VŠB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Applied Mathematics

Polyhedral Finite Elements in Heat transfer problems

Polyhedrální konečné prvky v úlohách vedení tepla

2020 Zbyšek Machaczek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Zadání bakalářské práce

Student:	**Zbyšek Machaczek**
Studijní program:	B2647 Informační a komunikační technologie
Studijní obor:	1103R031 Výpočetní matematika
Téma:	Polyhedrální konečné prvky v úlohách vedení tepla Polyhedral Finite Elements in Heat transfer problems
Jazyk vypracování:	čeština

Zásady pro vypracování:

Student nastuduje problematiku polyhedrálních konečných prvků v 2D a naimplementuje sestavovač pro matice tuhosti a pravé strany pro úlohy vedení tepla.

Budou uvažovány varianty pro Sibsonův a Lagrangeův interpolant násadových funkcí a také sestavení lokálních příspěvků na základě BEM úlohy pro element.

V experimentální části bude provedeno srovnání konvergence aproximace řešení pro tyto tři přístupy.

Seznam doporučené odborné literatury:

K. Hormann, N. Sukumar: Generalized Barycentric Coordinates in Computer Graphics and Computational Mechanics

Z. Jozefik, K. Chittepu, S. Kooshapur: Implementation of Polyhedral Finite Elements with application to heat transfer

N. Sukumar, J.E. Bolander: Voronoi-based Interpolants for Fracture Modelling

A. Gillette, A. Rand, Ch. Bajaj: Error Estimates for Generalized Barycentric Interpolation

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Oldřich Vlach, Ph.D.**

Datum zadání: 01.09.2019
Datum odevzdání: 30.04.2020

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

prof. RNDr. Jiří Bouchala, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

I hereby declare that this bachelor's thesis was written by myself. I have quoted all the references I have drawn upon

Albrechtice, May 8, 2020

Marianczak

I would like to thank my supervisor, Ing. Oldřich Vlach, Ph.D. for his guidance and for all the time he has spent helping me with this thesis. I would also like to thank my family for supporting me in my studies.

Abstrakt

Tato práce se primárně zabývá polyhedrální metodou konečných prvků ve 2D, pomocí které budeme řešit Poissonovu rovnici nabývající hodnoty nula na okraji. Při řešení polyhedrální metody konečných prvků se budeme muset zaměřit na zobecněné barycentrické souřadnice. Sekundárně se práce zabývá tvorbou efektivních kvadratur ve 2D, kde naimlementujeme algoritmus, založený na Newtonově metodě, který dokáže zefektivnit kvadraturu na vstupu.

Klíčová slova: MKP; polyhedrální konečné prvky; Poissonova rovnice; zobecněné barycentrické souřadnice; Wachspress souřadnice; meanvalue souřadnice; efektivní kvadratury

Abstract

The primary focus of this thesis is the polyhedral finite element method in 2D, using which we will be solving the Poisson equation valued zero on the boundary. While solving the polyhedral finite element method we will touch upon generalized barycentric coordinates. The secondary focus of this thesis is the generation of efficient quadratures, we will implement a Newton method based algorithm that can improve the efficiency of a given quadrature rule.

Keywords: FEM; polyhedral finite element method; Poisson equation; generalized barycentric coordinates; Wachspress coordinates; meanvalue coordinates; efficient quadratures

Contents

List of Figures

1 Intoduction

Partial differential equations (PDE's) allow us to model many phenomena in physics, however they do not always have an analytical solution, we must therefore make do with an approximate, aka numerical solution. The finite element method (FEM) is one of many methods used for solving PDE's numerically as we will discuss in Section 4. Without going into too much detail for now, the principle of FEM is finding the solution to the PDE's weak form in a finite–dimensional subspace of the PDE's solution space. Its signature feature is subdiving the domain of the PDE into more elements. This subdivision is called a mesh. The basis of the space we are trying to find an approximate solution in depends on this mesh.

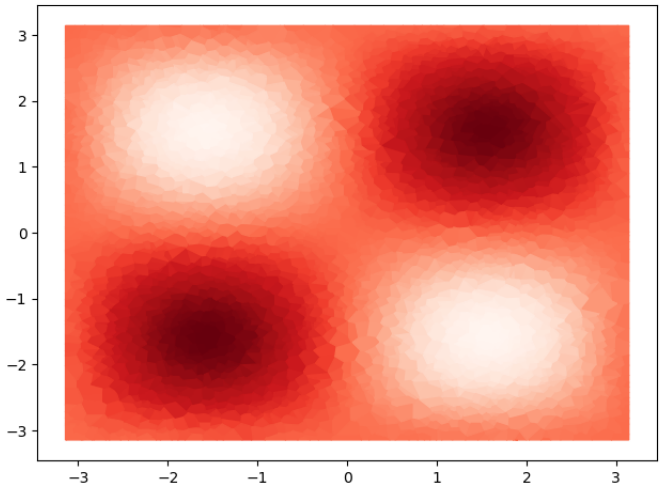

Figure 1.1: Plot of a numerical solution to a heat transfer problem.

To be able to perform the actual underlying computations we need to be able to integrate numerically. For this purpose we will be using the Gauss–Legendre quadrature. We will call a set of scalars (weights) and vectors (nodes) a quadrature if the dot product of function values of the nodes with the weights approximates the integral of the previously applied function over a given set. In 1D we can easily compute efficient quadratures, but in 2D it becomes rather difficult to produce quadrature rules with an arbitrary number of nodes. Thus we have decided to implement an algorithm introduced by Xiao and Gimbutas [1], that can compute a quadrature with near optimal efficiency. We will pass an initial quadrature to the algorithm, during each iteration this algorithm will then remove one quadrature node with the correspoding quadrature weight and then it will try to modify the other nodes and weights so that the new numerical integrant is equal to the original one. The modification of the quadrature nodes and weights will be done via the least squares Newton's method. The algorithm stops once Newton's method fails to converge. From our experience the algorithm nearly always stops when the number of quadrature nodes is near the optimal one.

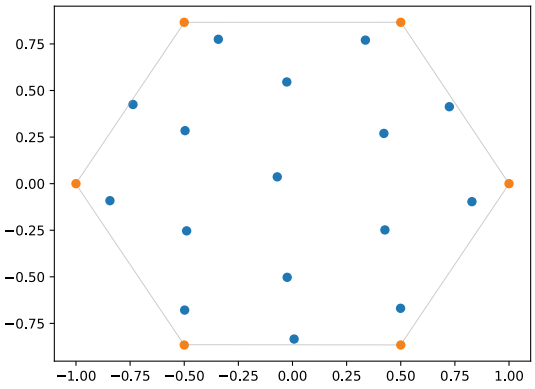

Figure 1.2: Nodes of a quadrature rule produced by the discussed algorithm.

2 Quadratures

The term quadrature or more precisely numerical quadrature is more or less synonymous to the term numerical integration. One of the simplest ways to integrate numerically is the so–called composite rectangle rule that can be immediately derived from the definition of a definite integral which is

$$\int_a^b f(x)\,\mathrm{d}x \equiv \lim_{n\to\infty} \sum_{i=1}^{n} f(\hat{x}_i)(x_{i+1} - x_i), \quad \hat{x}_i \in \langle x_i, x_{i+1}\rangle$$

we can approximate by making n finite, by choosing $\hat{x}_i$ to be the center of each interval we obtain

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=1}^{n} f(x_i)h, \quad h = \frac{b-a}{n},\ x_i = \frac{p_{i+1} + p_i}{2},\ p_i = a + (i-1)h.$$

We wrote this quadrature as a sum of function values in points x_i multiplied by some weights denoted as h. Weights do not have to be same for all i in general and they are usually denoted as w_i. All quadratures share this form, i.e., we can write all quadratures as $\boldsymbol{f} \cdot \boldsymbol{w}$ for some vectors $\boldsymbol{f}$ and $\boldsymbol{w}$ where $f_i = f(x_i),\ i = 1, 2, \ldots, n$.

Quadratures are absolutely key to the topic of this thesis – the finite element method which requires integrating over a very large number of sets, precisely over all elements in the mesh of our choice. For our computation to be of any use we require a certain amount of precision, and because the mesh usually consists of a very large number of elements, our second requirement for quadratures is computation speed. In this chapter we will define a quadrature type with the most suitable properties for our usages, expand the concept into 2–dimensional space and last but not least define a measure of quadrature's efficiency and introduce an algorithm that will improve the efficiency of the resulting quadratures.

2.1 Quadratures in 1D

There are many quadrature types, e.g., the Newton–Cotes quadrature where quadrature points are chosen equidistantly, or the Gauss–Chebyshev quadrature where we choose the roots of Chebyshev polynomials as quadrature points, but in this thesis, we will focus solely on the Gauss–Legendre quadrature.

Definition 1 (Gauss–Legendre quadrature)**.** Let

$$I^{GL}_{\langle -1,1\rangle}(f, n) = \sum_{i=1}^{n} f(x_i)w_i \approx \int_{-1}^{1} \omega(x)f(x)\,\mathrm{d}x \tag{2.1}$$

where x_i is the i–th root of the n–th degree Legendre polynomial (orthogonal in respect to the

inner product $\int_{-1}^{1} \omega(x) f(x) g(x)\,\mathrm{d}x$) and let

$$\begin{aligned} w_i &:= \int_{-1}^{1} \omega(x) L_i(x)\,\mathrm{d}x \\ L_i &:= \prod_{j=1, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \\ \omega(x) &= 1 \end{aligned}$$

then (2.1) is the n–th order Gauss Legendre quadrature approximately integrating function f on the interval $\langle -1, 1\rangle$.

Remark 2. The function $\omega : U \to \mathbb{R}$ in the definition above is called the weight function. A weight function must always satisfy

$$\forall x \in U :\ \omega(x) > 0$$

if U is not bounded then ω must further satisfy

$$\forall x \in U :\ \omega(x) e^{\alpha |x|} \leq c, \quad \alpha, c \in \mathbb{R}$$

As we could see for Legendre polynomials $\omega(x) = 1$ (which is the reason why we chose it over all the other types), but, e.g., for Chebyshev polynomials $\omega(x) = \frac{1}{\sqrt{1-x^2}}$.

Remark 3 (Change of interval). Changing the integration interval can be done in the following way

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{2} \int_{-1}^{1} f\left(\frac{b-a}{2} u + \frac{a+b}{2}\right) \mathrm{d}u$$

thus we define

$$I^{GL}_{\langle a,b\rangle}(f, n) := \frac{b-a}{2} \sum_{i=1}^{n} f\left(\frac{b-a}{2} u_i + \frac{a+b}{2}\right) w_i \tag{2.2}$$

where u_i is the i–th root of the n–th degree Legendre polynomial of the n–th order Gauss–Legendre quadrature approximating function $f(x)$ on the interval $\langle a, b\rangle$.

Theorem 4. *Alternative way to find values of* $\boldsymbol{w}$ *is to solve the system [1]*

$$\begin{bmatrix} \int_\Omega \omega(x)\phi_1(x)\,\mathrm{d}x \\ \int_\Omega \omega(x)\phi_2(x)\,\mathrm{d}x \\ \vdots \\ \int_\Omega \omega(x)\phi_m(x)\,\mathrm{d}x \end{bmatrix} = \begin{bmatrix} \phi_1(x_1) & \phi_1(x_2) & \dots & \phi_1(x_n) \\ \phi_2(x_1) & \phi_2(x_2) & \dots & \phi_2(x_n) \\ \vdots & & & \\ \phi_m(x_1) & \phi_m(x_2) & \dots & \phi_m(x_n) \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \tag{2.3}$$

Theorem 5 (Exactness of approximation)**.** *If n in the system (2.3) is greater or equal to $\frac{m}{2}$ then the resulting quadrature will be exact for any $\phi(x) \in \langle \phi_1, \phi_2, \ldots, \phi_m \rangle$, in the case that $n = \frac{m}{2}$ we call a 1D quadrature optimal. Let's take for example P^1_{2n-1},*

$$\left\langle P^1_{2n-1} \right\rangle = \left\langle x^0, x^1, \ldots, x^{2n-1} \right\rangle$$

the space of all polynomials of degree $2n-1$, we can say that

$$\forall p(x) \in P^1_{2n-1} : \int_a^b p(x)\,\mathrm{d}x = I^{GL}_{\langle a,b \rangle}(p, n) \tag{2.4}$$

Proof. Suppose $f \in P^1_{2n-1}$, we can write f as $p_n q + r$ where $\deg(p_n) \leq n, \deg(q) \leq \deg(r) \leq n-1$, because p_n is orthogonal to all polynomials of degree $n-1$ or less. It holds that

$$\int_a^b f(x)\,\mathrm{d}x = \int_a^b p_n(x)q(x)\,\mathrm{d}x + \int_a^b r(x)\,\mathrm{d}x = \int_a^b r(x)\,\mathrm{d}x$$

and because $r(x) = \sum_{i=0}^{n-1} r(x_i)L_i(x)$ and $p_n(x_i)q(x_i) + r(x_i) = r(x_i)$ we can say that

$$\int_a^b r(x)\,\mathrm{d}x = \sum_{i=0}^{n-1} f(x_i) \int_a^b L_i(x)\,\mathrm{d}x = \sum_{i=0}^{n-1} f(x_i) w_i\,\mathrm{d}x$$

prooving the statement above. □

2.2 Quadratures in 2D

While the concept of quadratures stays the same in 2D, we must carefully consider what it means for a quadrature to be efficient and how to compose them if they are to integrate over more tricky regions. Once we have a quadrature rule that integrates over a polygon with $v \in \mathbb{N} : \ v \geq 3$ vertices we can relatively easily transform it into a rule that integrates over a different polygon with v vertices using generalized barycentric coordinates, we will introduce those in Chapter 3.

Definition 6 (Quadrature in 2D)**.** The expression

$$\sum_{i=1}^{n} f(\boldsymbol{x}_i) w_i, \quad \boldsymbol{x}_i \in \mathbb{R}^2 \tag{2.5}$$

is called a 2–dimensional quadrature rule if $w_1, w_2, \ldots, w_n$ satisfy (2.3) with $\Omega \subset \mathbb{R}^2$ being a bounded closed set over which we want to integrate and $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ being Gauss points such that $\boldsymbol{x}_i \in \Omega \subset \mathbb{R}^2$, $i = 1, 2, \ldots, n$.

We will discuss the exact choice of Gauss points later on.

Definition 7 (Efficiency of a 2D quadrature)**.** Let

$$E = \frac{\frac{m}{d+1}}{n} \tag{2.6}$$

where m, n are m, n from (2.3) and d is the dimension of the Euclidean space the reference elements are embedded in, this gives us $E = \frac{m/3}{n}$ for 2D quadrature rules, we call a quadrature rule efficient if E is close to one, see Xiao and Gimbutas [1].

2.2.1 Initial quadrature over a rectangle

Let

$$\boldsymbol{x} = \boldsymbol{u} \times \boldsymbol{v}$$

where $\boldsymbol{u}$ is a vector of roots of the o–th order Legendre polynomial stretched to the interval $\langle a, b\rangle$ and $\boldsymbol{v}$ is a vector of roots of the o–th order Legendre polynomial stretched to the interval $\langle c, d\rangle$, we can compose the o^2 point quadrature rule over the rectangle $\langle a, b\rangle \times \langle c, d\rangle$ with Gauss points given by $\boldsymbol{x}$. We can see the result in Figure 2.1. Another thing the figure shows us is the distribution of Legendre polynomial roots

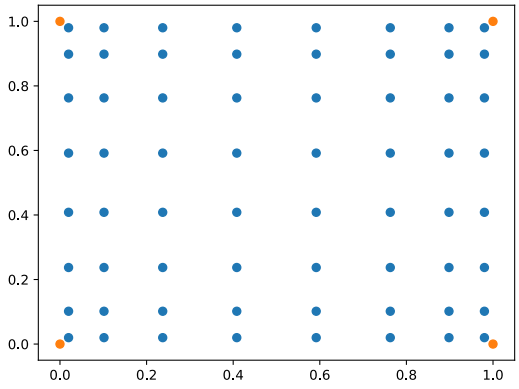

Figure 2.1: A tensor product rectangle rule of order 8.

Remark 8 (Alternative formula for the weights). If we want to find the weights of a quadrature over a rectangle, it is not necessary to solve the system (2.3), instead we can use the following formula

$$\boldsymbol{w} = \boldsymbol{w}_x \otimes \boldsymbol{w}_y \tag{2.7}$$

where $\boldsymbol{w}_x$ are weights corresponding to the points $\boldsymbol{u}$. $\boldsymbol{w}_y$ are weights corresponding to the points $\boldsymbol{v}$ and $\otimes$ is the tensor product. This is why quadrature rules produced in this manner are called tensor product rules.

2.2.2 Initial quadrature over a triangle

Let $\triangle$ be the triangle with vertices $(0, 0)$, $(1, 0)$ and $(0, 1)$

$$\iint_{\triangle} f(x,y)\,\mathrm{d}x\mathrm{d}y = \int_0^1 \int_0^{1-x} f(x,y)\,\mathrm{d}x\mathrm{d}y \tag{2.8}$$

We can use the substitution

$$x = u, \quad y = (1-u)v$$

with the determinant of its Jacobian

$$\begin{vmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ -v & 1-u \end{vmatrix} = 1-u$$

by applying the substitution we get

$$\int_0^1 \int_0^{1-x} f(x,y)\,\mathrm{d}x\mathrm{d}y = \int_0^1 \int_0^1 (1-u) f(u,(1-u)v)\,\mathrm{d}x\mathrm{d}y$$

which means that if we take the quadrature nodes of the square

$$\square := \langle 0,1\rangle \times \langle 0,1\rangle$$

where $\boldsymbol{x}_\square = \boldsymbol{u}_\square \times \boldsymbol{v}_\square$ are tensor product Gauss points and $\boldsymbol{w}_\square$ are the corresponding weights, then

$$\boldsymbol{x}_\triangle = \boldsymbol{u}_\square \times (1 - \boldsymbol{u}_\square) \odot \boldsymbol{v}_\square \tag{2.9}$$

and

$$\boldsymbol{w}_\triangle = (1 - \boldsymbol{u}_\square) \odot \boldsymbol{w}_\square \tag{2.10}$$

where $\odot$ is elementwise multiplication give us a quadrature rule over $\triangle$. The weights can alternatively be computed using the formula (2.3).

Remark 9 (Barycentric coordinates). We will discuss barycentric coordinates more deeply later on. For now, we would only like to provide the reader with a method to transform the initial triangle quadrature rule into a quadrature rule for a triangle with one vertex in the point $(0,0)$. This is necessary for creating initial quadratures for higher order polygons.

Let $\triangle$ be the triangle given by vertices $(0,0)$, $(1,0)$, $(0,1)$ and let T be the triangle given by vertices $(0,0)$, $(b1,b2)$, $(c1,c2)$ it holds that

$$T = \{(x_\triangle b1 + y_\triangle c1, x_\triangle b2 + y_\triangle c2) \mid (x_\triangle, y_\triangle) \in \triangle\}$$

2.2.3 Initial quadrature over a convex polygon

To compose a quadrature over a convex polygon with n vertices all we have to do is to separate the polygon into n triangles, all of which have a common vertex in the center of the polygon

and save nodes of all the resulting triangles.

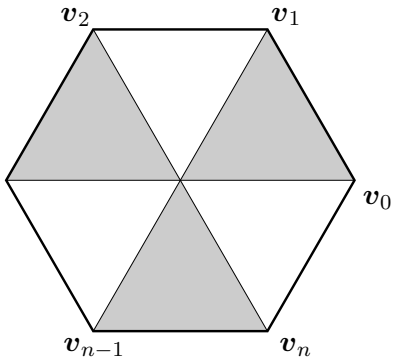

Figure 2.2: Triangle decomposition of a polygon.

There are obviously other ways to decompose polygons, e.g., the hexagon in the figure above could be separated into two quadrilaterals, however, the method we use works for all convex polygons.

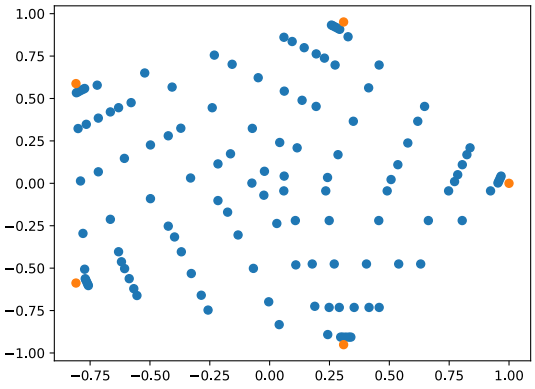

Figure 2.3: Tensor product pentagon rule.

2.3 Construction of efficient quadratures

Reflecting on the methods for creating initial quadrature rules above, we can see that they can only give us n^2 point quadrature rules. For polygons of degree 5 and higher it is even worse, we can only produce vn^2 point quadrature rules where v is the number of vertices the chosen polygon has. Our fundamental goal is to create quadrature rules for which it holds that $E = \frac{m/3}{n}$ is close to 1.

We will only focus on integrating polynomials in this paper, therefore from (2.3) we will choose $\phi_1, \phi_2, \ldots, \phi_m$ such that $\langle \phi_1, \phi_2, \ldots, \phi_m \rangle = P_k$ where P_k is the space of all functions in the form $\sum_{l=1}^{m} a_l x^{\eta_l} y^{\theta_l} = \sum_{i=0}^{k} \sum_{j=0}^{i} a_{i,j} x^{i-j} y^j$, i.e., the space of all two variable polynomials of

degree k or less. The basis of P_k will contain

$$\sum_{i=0}^{k}\sum_{j=0}^{i} 1 = (k+1)(k+2)/2$$

basis functions.

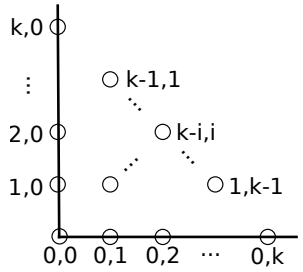

Figure 2.4: The basis of P_k.

The method we use, taken from [1], works on the principle of omitting the point with the smallest significance index (together with the correspoding weight) and subsequently using the least squares Newton's method with the system (2.3) on the remaining points to try and find $n-1$ quadrature points and weights that fulfill (2.3). This process is repeated for as long as Newton's method keeps converging.

Definition 10 (Newton's method)**.** The least squares Newton's method is used to solve the equation

$$\boldsymbol{F}(\boldsymbol{X}) := \begin{pmatrix} f_1(\boldsymbol{X}) \\ f_2(\boldsymbol{X}) \\ \vdots \\ f_m(\boldsymbol{X}) \end{pmatrix} = \boldsymbol{0} \tag{2.11}$$

where $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^m, n \geq m$. The iteration step is

$$\boldsymbol{X}^{k+1} := \boldsymbol{X}^k - \mathsf{J}^{\dagger}(\boldsymbol{X}^k)\boldsymbol{F}(\boldsymbol{X}^k) \tag{2.12}$$

where $k > 0$, the initial approximation $\boldsymbol{X}^0$ is given and $\mathsf{J}^{\dagger}$ is the pseudo inverse of

$$\mathsf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{X}) & \frac{\partial f_1}{\partial x_2}(\boldsymbol{X}) & \dots & \frac{\partial f_1}{\partial x_n}(\boldsymbol{X}) \\ \frac{\partial f_2}{\partial x_1}(\boldsymbol{X}) & \frac{\partial f_2}{\partial x_2}(\boldsymbol{X}) & \dots & \frac{\partial f_2}{\partial x_n}(\boldsymbol{X}) \\ \vdots & & & \\ \frac{\partial f_m}{\partial x_1}(\boldsymbol{X}) & \frac{\partial f_m}{\partial x_2}(\boldsymbol{X}) & \dots & \frac{\partial f_m}{\partial x_n}(\boldsymbol{X}) \end{bmatrix} \tag{2.13}$$

Remark 11 (Orthonormal bases). The monomial basis of P_k might at first seem as an obvious choice for $\phi_1, \phi_2, \dots, \phi_m$, however monomials tend to be poorly conditioned. Therefore we will instead use polynomials orthonormal with respect to the inner product $\int_\Omega f(x,y)g(x,y)\,\mathrm{d}x\mathrm{d}y$,

where Ω is a polygonal element in our case. The reference quadrature integrates over and $f, g \in \langle P_k \rangle$.

Algorithm 12 (Node elimination)**.** *Given a set of m functions $\phi_1, \phi_2, \ldots, \phi_m$ and a n point quadrature rule, the algorithm returns a p point quadrature rule for integrating $\phi_1, \phi_2, \ldots, \phi_m$ where $p \leq n$. We will refer to the array containing quadrature points as $\boldsymbol{x}$, to the array containing quadrature weights as $\boldsymbol{w}$ and we will define $\boldsymbol{X} = \begin{bmatrix} x_1 & \cdots & x_m & y_1 & \cdots & y_m & w_1 & \cdots & w_m \end{bmatrix}^\top$.*

---

while the Newton's method converges
  reorder x and w according to the significance index in increasing order
  $\boldsymbol{Y} = \begin{bmatrix} x_2 & \cdots & x_m & y_2 & \cdots & y_m & w_2 & \cdots & w_m \end{bmatrix}^\top$
  run Newton's method for $\boldsymbol{Y}$
  if Newton's method succeeds
    $\boldsymbol{X} = \boldsymbol{Y}$
return $\boldsymbol{X}$

The significance index of $x_j \in \boldsymbol{x}, w_j \in \boldsymbol{w}$ is

$$s_j := w_j \sum_{i=1}^{m} \phi_i^2(\boldsymbol{x}_j)$$

Remark 13 (Points outside the reference elements). It is possible for points to jump outside the boundary of reference elements during the execution of node elimination, however, it does not happen often and if the points do jump outside the boundary it is not by much. A good way to deal with this problem is to assign a large negative significance index to a point outside the reference element. From our experience this still does not prevent this phenomenon completely, so some experimentation with the initial approximation for the Newton's method might be necessary. We should mention that the authors of the node elimination algorithm have proposed an algorithm for constructing initial approximations for the Newton's method using a pivotized Gram–Schmidt process, but it proved to be unnecessary since it is not a problem to get the algorithm to converge. It is also noteworthy that the algorithm is not very sensitive to the exact form of s_j, so during the reordering we can appose other specific requirements, such as symmetry, positivity of weights and so on. The form we are using tends to produce quadratures with positive weights, if one is not interested in positive weights, they can instead use $s_j := \sum_{i=1}^{m} \phi_i^2(\boldsymbol{x}_j)$.

2.3.1 Application of Newton's method

Let

$$\boldsymbol{F} \in C^\infty(\mathbb{R}^{3n}, \mathbb{R}^m), \quad \boldsymbol{\chi} = \begin{bmatrix} x_1 & \cdots & x_n & y_1 & \cdots & y_n & w_1 & \cdots & w_n \end{bmatrix}^\top \in \mathbb{R}^{3n}$$
$$\boldsymbol{\gamma} = \begin{bmatrix} x & y \end{bmatrix}^\top \in \mathbb{R}^2, \quad \boldsymbol{\gamma}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^\top \in \mathbb{R}^2, \quad \boldsymbol{w} \in \mathbb{R}^n, \quad \Omega \subset \mathbb{R}^2$$

$$\phi_j(\boldsymbol{\gamma}) = \sum_{l=1}^{m} \alpha_{jl} x^{\eta_l} y^{\theta_l}$$

where

$$\boldsymbol{F}(\boldsymbol{\chi}) \equiv \begin{bmatrix} \phi_1(\boldsymbol{\gamma}_1) & \cdots & \phi_1(\boldsymbol{\gamma}_n) \\ \vdots & \ddots & \vdots \\ \phi_m(\boldsymbol{\gamma}_1) & \cdots & \phi_m(\boldsymbol{\gamma}_n) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix} - \begin{bmatrix} I_1 \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad I_j = \int_\Omega \omega(\boldsymbol{\gamma}) \phi_j(\boldsymbol{\gamma}) \, \mathrm{d}\boldsymbol{\gamma}$$
$$f_j(\boldsymbol{\chi}) = \sum_{k=1}^{n} \phi_j(\boldsymbol{\gamma}_k) w_k - I_j = \sum_{k=1}^{n} \sum_{l=1}^{m} \alpha_{jl} x_k^{\eta_l} y_k^{\theta_l} w_k - I_j$$

and for $\mathsf{J} \in C^\infty(\mathbb{R}^{3n}, \mathbb{R}^{m \times 3n})$ holds

$$\mathsf{J}(\boldsymbol{\chi}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{\gamma}_1) & \cdots & \frac{\partial f_1}{\partial x_m}(\boldsymbol{\gamma}_n) & \frac{\partial f_1}{\partial y_1}(\boldsymbol{\gamma}_1) & \cdots & \frac{\partial f_1}{\partial y_m}(\boldsymbol{\gamma}_n) & \frac{\partial f_1}{\partial w_1}(\boldsymbol{\gamma}_1) & \cdots & \frac{\partial f_1}{\partial w_n}(\boldsymbol{\gamma}_n) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\boldsymbol{\gamma}_1) & \cdots & \frac{\partial f_m}{\partial x_n}(\boldsymbol{\gamma}_n) & \frac{\partial f_m}{\partial y_1}(\boldsymbol{\gamma}_1) & \cdots & \frac{\partial f_m}{\partial y_n}(\boldsymbol{\gamma}_n) & \frac{\partial f_m}{\partial w_1}(\boldsymbol{\gamma}_1) & \cdots & \frac{\partial f_m}{\partial w_n}(\boldsymbol{\gamma}_n) \end{bmatrix}$$
$$= \begin{bmatrix} \mathsf{B} & \mathsf{C} & \mathsf{A} \end{bmatrix}$$

where $\mathsf{B}, \mathsf{C}, \mathsf{A} \in \mathbb{R}^{m \times n}$ and

$$\frac{\partial f_j}{\partial x_i} = \sum_{l=1}^{m} \alpha_{jl} \eta_l x_i^{\eta_l - 1} y_i^{\theta_l} w_i \equiv B_{ji}$$
$$\frac{\partial f_j}{\partial y_i} = \sum_{l=1}^{m} \alpha_{jl} \theta_l x_i^{\eta_l} y_i^{\theta_l - 1} w_i \equiv C_{ji}$$
$$\frac{\partial f_j}{\partial w_i} = \sum_{l=1}^{m} \alpha_{jl} x_i^{\eta_l} y_i^{\theta_l} \equiv A_{ji}$$

It is obvious that Newton's method solving $\boldsymbol{F}(\boldsymbol{\chi}) = \boldsymbol{0}$ will produce a quadrature rule for integrating $\phi_1, \phi_2, \ldots, \phi_m$ if it converges.

Remark 14 (Convergence of Newton's method). Generally speaking, Newton's method will only converge when the initial approximation is close to the final solution. From our observations it is not a problem to get the algorithm to converge, but using a number of points considerably larger that of the expected result does not seem to be a good choice as the points are more likely to jump outside the reference element. We have computed efficient quadrature rules for integrating $P_3, P_4, \ldots, P_{10}$ over polygons with up to 10 vertices. From [1] we can see that the

algorithm **??** is well capable of producing quadrature rules for integrating polynomials of a much higher order.

2.4 Results

We have manufactured quadrature rules for integrating functions from $P_3, \ldots, P_{10}$ over polygons with up to 10 vertices, this is more than sufficient for our purposes. The algorithm nearly always managed to get very close to the expected number of points, i.e., such that $E = \frac{m/3}{n}$ is close to one, although sometimes it produces degenerate quadratures with lower efficiency coefficients. Such quadratures typically have a bunch of points clumped together, in such case using a different initial quadrature may be necessary. It goes without saying that our final quadratures have all its points inside the reference elements, the alternative is undesirable for our purposes. Here are some quadrature rules produced by the node elimination algorithm. For convenience we will be labeling a quadrature rule for integrating in P_k over a polygon with v vertices as $Q(v, k)$

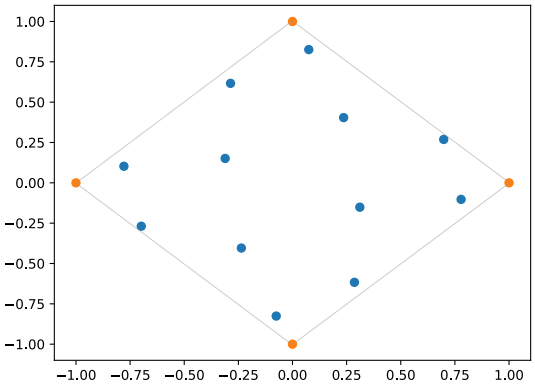

Figure 2.5: The rectangle rule for integrating in P_7 ($Q(4, 7)$).

We can see that $Q(4, 7)$ consists of 12 points which means its efficiency is $E = \frac{\frac{(7+1)(7+2)/2}{3}}{12} = 1$, that leaves us very satisfied.

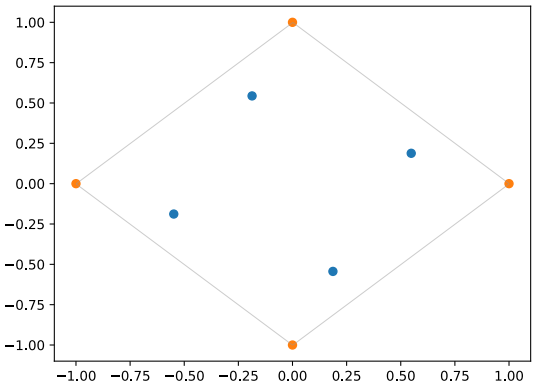

Figure 2.6: $Q(4,3)$.

Looking at a much lower order rule such as $Q(4,3)$ we can see that the points remain nicely distributed, that is actually a very important feature we are looking for, its efficiency is also 1.

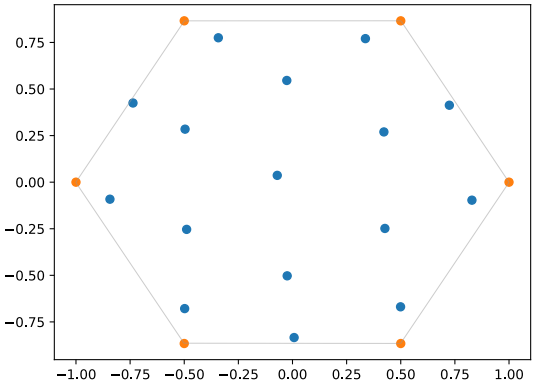

Figure 2.7: $Q(6,8)$, $E_{Q(6,8)} = 0.9375$.

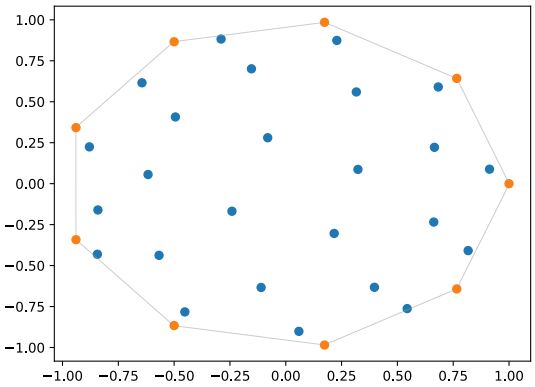

Figure 2.8: $Q(9, 10)$, $E_{Q(9,10)} = 0.88$.

3 Generalized barycentric coordinates

Generalized barycentric coordinates (GBC's) are a generalization of barycentric coordinates, they are another key component of the polygonal finite element method, they have two uses that are of interest to us. The first one brings us back to Chapter 2 where we were constructing reference quadratures. GBC's make it possible for us to map the reference quadratures onto certain non–reference sets (see Definiton 17). The second usage is in the isoparametric approach to the polygonal FEM framework (which will be discussed in Chapter 4). So in this section we will discuss requirements on GBC's and their basic properties, define 3 types of GBC's and describe the attributes of each type. Note that in this chapter we will index from zero.

3.1 What are GBCs

Definition 15 (Generalized Barycentric coordinates(GBC's))**.** Let an open set $\mathscr{P} \subset \mathbb{R}^2$ represent a convex polygon with vertices $\boldsymbol{v}_0, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{n-1}$, $n \geq 3$ in an anticlockwise ordering. A set of functions $N_i : \mathscr{P} \to \mathbb{R}$, $i = 0, \ldots, n-1$ will be called generalized barycentric coordinates, if it for all i and for all $\boldsymbol{v} \in \mathscr{P}$ satisfies (see Floater [3])

$$N_i(\boldsymbol{v}) \geq 0, \quad \sum_{i=0}^{n-1} N_i(\boldsymbol{v}) = 1, \quad \sum_{i=0}^{n-1} N_i(\boldsymbol{v})\boldsymbol{v}_i = \boldsymbol{v} \tag{3.1}$$

Remark 16 (Properties of GBCs). For $n = 3$ the GBC are uniquely determined as

$$N_i(\boldsymbol{v}) = \frac{A(\boldsymbol{v}, \boldsymbol{v}_{i+1}, \boldsymbol{v}_{i+2})}{A(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3)}, \quad i = 0, 1, 2 \tag{3.2}$$

where $A(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ is the signed area of the triangle given by vertices $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$. A reader who has come across barycentric coordinates before will notice that 3.2 simply denotes standart barycentric coordinates [3].

Remark. For $n \geq 4$ the GBC are not uniquely determined, in fact there are many types. They all share the following properties as described in Floater [2]:

1. The functions N_i have a unique continuous extention to $\partial\mathscr{P}$, the boundary of $\mathscr{P}$.
2. Lagrange property: $N_i(v_j) = \delta_{ij}$.
3. Piecewise linearity on $\partial\mathscr{P}$:

$$N_i((1-\mu)\boldsymbol{v}_j + \mu\boldsymbol{v}_{j+1}) = (1-\mu)N_i(\boldsymbol{v}_j) + \mu N_i(\boldsymbol{v}_{j+1}), \quad \mu \in (0,1)$$

4. Barycentric interpolation: if

$$g(\boldsymbol{v}) = \sum_{i=0}^{n-1} N_i(\boldsymbol{v}) f(\boldsymbol{v}_i), \quad \boldsymbol{v} \in \mathscr{P}$$

then $f(\boldsymbol{v}_i) = g(\boldsymbol{v}_i)$ and we call g a barycentric interpolant of f.

5. Linear precision: if f is linear, then $g = f$.

6. $\vartheta_i(\boldsymbol{v}) \leq N_i(\boldsymbol{v}) \leq \theta_i(\boldsymbol{v})$ where $\boldsymbol{v} \in \mathscr{P}$ and $\theta_i, \vartheta_i : \mathscr{P} \to \mathbb{R}$ are continuous, piecewise linear functions over the partitions of $\mathscr{P}$ shown in Figure **??** satisfying $\theta_i(\boldsymbol{v}_j) = \vartheta_i(\boldsymbol{v}_j) = \delta_{ij}$. θ_i is the least upper bound and ϑ_i is the greatest lower bound of N_i.

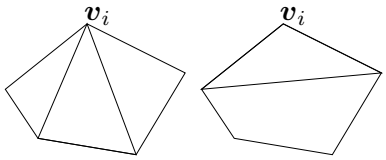

Figure 3.1: Partitions of L_i and ϑ_i.

3.1.1 Definitions for each type

Let $\mathscr{P}$ be a polygon with vertices $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n-1}$ depicted in 3.2 and let $\boldsymbol{v} \in \mathscr{P}$. We denote

$$\boldsymbol{s}_i(\boldsymbol{v}) = \boldsymbol{v}_i - \boldsymbol{v} \qquad r_i(\boldsymbol{v}) = \|\boldsymbol{s}_i(\boldsymbol{v})\| \qquad \boldsymbol{e}_i(\boldsymbol{v}) = \left(r_i^{-1}\boldsymbol{s}_i\right)(\boldsymbol{v})$$

$$\boldsymbol{l}_i = \boldsymbol{v}_{i+1} - \boldsymbol{v}_i \qquad \boldsymbol{n}_i = \mathrm{rot}_{90}(\boldsymbol{l}_i) = \begin{bmatrix} -l_{i[1]} \\ l_{i[0]} \end{bmatrix} \qquad h_i(\boldsymbol{v}) = \boldsymbol{s}_i(\boldsymbol{v}) \cdot \boldsymbol{n}_i$$

$$A_i(\boldsymbol{v}) = \frac{1}{2}\left(\begin{bmatrix} v_{i[0]} - v_{[0]} \\ v_{i[1]} - v_{[1]} \\ 0 \end{bmatrix} \times \begin{bmatrix} v_{(i+1)[0]} - v_{[0]} \\ v_{(i+1)[1]} - v_{[1]} \\ 0 \end{bmatrix}\right)_{[2]}, \quad A_i \equiv A_i(\boldsymbol{v})$$

$$B_i(\boldsymbol{v}) = \frac{1}{2}\left(\begin{bmatrix} v_{(i-1)[0]} - v_{[0]} \\ v_{(i-1)[1]} - v_{[1]} \\ 0 \end{bmatrix} \times \begin{bmatrix} v_{(i+1)[0]} - v_{[0]} \\ v_{(i+1)[1]} - v_{[1]} \\ 0 \end{bmatrix}\right)_{[2]}, \quad B_i \equiv B_i(\boldsymbol{v})$$

$$C_i = \frac{1}{2}\left(\begin{bmatrix} v_{(i+1)[0]} - v_{i[0]} \\ v_{(i+1)[1]} - v_{i[1]} \\ 0 \end{bmatrix} \times \begin{bmatrix} v_{(i-1)[0]} - v_{i[0]} \\ v_{(i-1)[1]} - v_{i[1]} \\ 0 \end{bmatrix}\right)_{[2]}$$

$$D_i(\boldsymbol{v}) = \begin{bmatrix} v_{i[0]} - v_{[0]} \\ v_{i[1]} - v_{[1]} \end{bmatrix} \cdot \begin{bmatrix} v_{(i+1)[0]} - v_{[0]} \\ v_{(i+1)[1]} - v_{[1]} \end{bmatrix}, \quad D_i \equiv D_i(\boldsymbol{v})$$

$$\cos(\alpha_i) = \boldsymbol{e}_i \cdot \boldsymbol{e}_{i+1}$$

$$\sin(\alpha_i) = \boldsymbol{e}_i \times \boldsymbol{e}_{i+1}$$

$$t_i = \tan\left(\frac{\alpha_i}{2}\right) = \frac{r_{i+1} - \frac{D_i}{r_i}}{A_i} =$$

$$= \frac{\sin(\alpha_i)}{1 + \cos(\alpha_i)} = \frac{\boldsymbol{e}_i \times \boldsymbol{e}_{i+1}}{1 + \boldsymbol{e}_i \cdot \boldsymbol{e}_{i+1}}$$

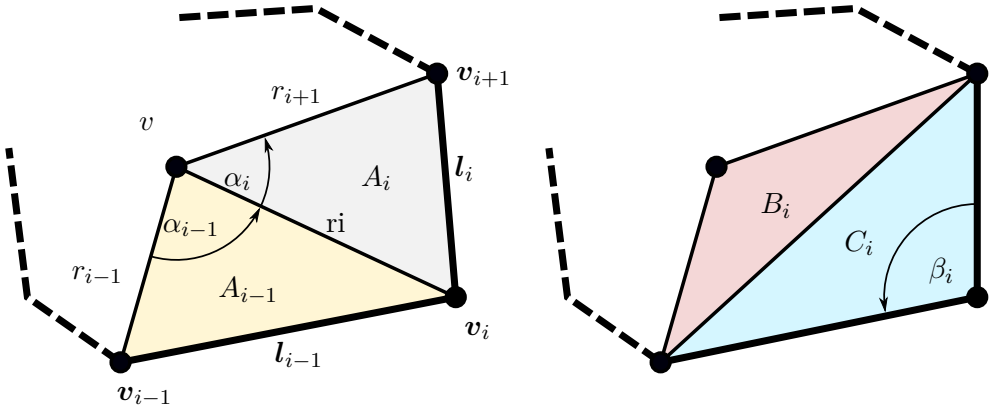

Figure 3.2: Polygon with n vertices - notation.

Let

$$N_i(\boldsymbol{v}) = \frac{w_i}{\sum_{j=0}^{n-1} w_j} \tag{3.3}$$

where

$$w_i(\boldsymbol{v}) = \frac{f(r_{i+1})A_{i-1} - f(r_i)B_i + f(r_{i-1})A_i}{A_{i-1}A_i}, \quad w_i \equiv w_i(\boldsymbol{v})$$

we define each coordinate type with $f(r) \in \{r^0, r^1, r^2\}$ as

$$\text{Wachspress} \quad w_i^{WP} = \frac{r_{i+1}^0 A_{i-1} - r_i^0 B_i + r_{i-1}^0 A_i}{A_{i-1}A_i} = \frac{C_i}{A_{i-1}A_i} = 2\frac{\boldsymbol{n}_{i-1} \times \boldsymbol{n}_i}{h_{i-1}h_i} \tag{3.4}$$

$$\text{Mean value} \quad w_i^{MV} = \frac{r_{i+1}A_{i-1} - r_i B_i + r_{i-1}A_i}{A_{i-1}A_i} = \frac{t_{i-1} + t_i}{r_i} \tag{3.5}$$

$$\text{Discrete harmonic} \quad w_i^{DH} = \frac{r_{i+1}^2 A_{i-1} - r_i^2 B_i + r_{i-1}^2 A_i}{A_{i-1}A_i} = 2(\cot(\gamma_i) + \cot(\delta_{i-1})) \tag{3.6}$$

where γ_i is the angle in the triangle with vertices $\boldsymbol{v}, \boldsymbol{v}_1, \boldsymbol{v}_{i+1}$ at the corner $\boldsymbol{v}_{i+1}$ and δ_{i-1} is the angle in the triangle with vertices $\boldsymbol{v}, \boldsymbol{v}_{i-1}, \boldsymbol{v}_i$ at the corner $\boldsymbol{v}_{i-1}$. All of those are what we defined as GBC, but we are not done yet, we still need to find their derivatives.

Let

$$\nabla(\cdot) \equiv \begin{bmatrix} \partial_{v_{[0]}} \\ \partial_{v_{[1]}} \end{bmatrix} (\cdot)$$

it holds

$$\nabla s_{i[0]} = \nabla\left(v_{i[0]} - v_{[0]}\right) = \begin{bmatrix}-1\\0\end{bmatrix} \qquad \nabla s_{i[1]} = \nabla\left(v_{i[1]} - v_{[1]}\right) = \begin{bmatrix}0\\-1\end{bmatrix}$$

$$\nabla r_i = \nabla\left(\sqrt{\left(s_{i[0]}\right)^2 + \left(s_{i[1]}\right)^2}\right) = \frac{1}{2r_i}\left(2s_{i[0]}\nabla s_{i[0]} + 2s_{i[1]}\nabla s_{i[1]}\right) = \frac{-1}{r_i}\boldsymbol{s}_i = -\boldsymbol{e}_i$$

$$\nabla h_i = \nabla\left(\boldsymbol{s}_i \cdot \boldsymbol{n}_i\right) = -\boldsymbol{n}_i$$

$$\nabla A_i = \nabla\left(\frac{1}{2}s_{i[0]}s_{(i+1)[1]} - \frac{1}{2}s_{i[1]}s_{(i+1)[0]}\right) = \frac{1}{2}\begin{bmatrix}-s_{(i+1)[1]} + s_{i[1]}\\-s_{i[0]} + s_{(i+1)[0]}\end{bmatrix} = \frac{1}{2}\begin{bmatrix}-l_{i[1]}\\l_{i[0]}\end{bmatrix}$$

$$\nabla B_i = \nabla\left(\frac{1}{2}\left(v_{(i-1)[0]} - v_{[0]}\right)\left(v_{(i+1)[1]} - v_{[1]}\right) - \frac{1}{2}\left(v_{(i-1)[1]} - v_{[1]}\right)\left(v_{(i+1)[0]} - v_{[0]}\right)\right) =$$

$$= \frac{1}{2}\begin{bmatrix}v_{(i-1)[1]} - v_{(i+1)[1]}\\v_{(i+1)[0]} - v_{(i-1)[0]}\end{bmatrix}$$

$$\nabla C_i = \boldsymbol{0}$$

$$\nabla D_i = \nabla\left(s_{i[0]}s_{(i+1)[0]} + s_{i[1]}s_{(i+1)[1]}\right) = 2\boldsymbol{v} - \boldsymbol{v}_i - \boldsymbol{v}_{i+1}$$

$$\nabla\cos(\alpha_i) = \nabla\left(\boldsymbol{e}_i \cdot \boldsymbol{e}_{i+1}\right) = -\left(\boldsymbol{e}_i \times \boldsymbol{e}_{i+1}\right)\boldsymbol{c}_i^{\perp} = -\sin(\alpha_i)\boldsymbol{c}_i^{\perp}$$

$$\nabla\sin(\alpha_i) = \nabla\left(\boldsymbol{e}_i \times \boldsymbol{e}_{i+1}\right) = \left(\boldsymbol{e}_i \cdot \boldsymbol{e}_{i+1}\right)\boldsymbol{c}_i^{\perp} = \cos(\alpha_i)\boldsymbol{c}_i^{\perp}$$

$$\nabla t_i = \nabla\left(\frac{\sin(\alpha_i)}{1+\cos(\alpha_i)}\right) = \frac{\cos(\alpha_i)\left(1+\cos(\alpha_i)\right) - \sin(\alpha_i)\left(-\sin(\alpha_i)\right)}{\left(1+\cos(\alpha_i)\right)^2}\boldsymbol{c}_i^{\perp} =$$

$$= \frac{1+\cos(\alpha_i)}{\left(1+\cos(\alpha_i)\right)^2}\boldsymbol{c}_i^{\perp} = \frac{1}{1+\cos(\alpha_i)}\boldsymbol{c}_i^{\perp} = \frac{t_i}{\sin(\alpha_i)}\boldsymbol{c}_i^{\perp}$$

because

$$\nabla e_{i[0]} = \nabla\frac{s_{i[0]}}{r_i} = \frac{r_i\nabla s_{i[0]} - s_{i[0]}\nabla r_i}{r_i^2} = \frac{1}{r_i}\begin{bmatrix}-1\\0\end{bmatrix} + \frac{s_{i[0]}}{r_i^3}\boldsymbol{s}_i =$$

$$= \frac{1}{r_i}\left(\begin{bmatrix}-1\\0\end{bmatrix} + e_{i[0]}\boldsymbol{e}_i\right) = \frac{1}{r_i}\begin{bmatrix}-1 + e_{i[0]}e_{i[0]}\\e_{i[0]}e_{i[1]}\end{bmatrix} = \frac{1}{r_i}\begin{bmatrix}-e_{i[1]}e_{i[1]}\\e_{i[0]}e_{i[1]}\end{bmatrix} =$$

$$\frac{e_{i[1]}}{r_i}\begin{bmatrix}-e_{i[1]}\\e_{i[0]}\end{bmatrix} = \frac{e_{i[1]}}{r_i}\boldsymbol{e}_i^{\perp}$$

$$\nabla e_{i[1]} = \frac{1}{r_i}\begin{bmatrix}0\\-1\end{bmatrix} + \frac{s_{i[1]}}{r_i^3}\boldsymbol{s}_i = \frac{1}{r_i}\begin{bmatrix}e_{i[1]}e_{i[0]}\\-1 + e_{i[1]}e_{i[1]}\end{bmatrix} = \frac{1}{r_i}\begin{bmatrix}e_{i[0]}e_{i[1]}\\-e_{i[0]}e_{i[0]}\end{bmatrix} = -\frac{e_{i[0]}}{r_i}\boldsymbol{c}_i^{\perp}$$

and

$$\begin{aligned}
\nabla\left(\boldsymbol{e}_i\cdot\boldsymbol{e}_{i+1}\right) &= e_{(i+1)[0]}\nabla e_{i[0]} + e_{i[0]}\nabla e_{(i+1)[0]} + e_{(i+1)[1]}\nabla e_{i[1]} + e_{i[1]}\nabla e_{(i+1)[1]} \\
&= e_{(i+1)[0]}\frac{e_{i[1]}}{r_i}\boldsymbol{e}_i^{\perp} + e_{i[0]}\frac{e_{(i+1)[1]}}{r_{i+1}}\boldsymbol{e}_{i+1}^{\perp} - e_{(i+1)[1]}\frac{e_{i[0]}}{r_i}\boldsymbol{e}_i^{\perp} - e_{i[1]}\frac{e_{(i+1)[0]}}{r_{i+1}}\boldsymbol{e}_{i+1}^{\perp} \\
&= -e_{i[0]}e_{(i+1)[1]}\left(\frac{1}{r_i}\boldsymbol{e}_i^{\perp} - \frac{1}{r_{i+1}}\boldsymbol{e}_{i+1}^{\perp}\right) + e_{i[1]}e_{(i+1)[0]}\left(\frac{1}{r_i}\boldsymbol{e}_i^{\perp} - \frac{1}{r_{i+1}}\boldsymbol{e}_{i+1}^{\perp}\right) \\
&= -\left(e_{i[0]}e_{(i+1)[1]} - e_{i[1]}e_{(i+1)[0]}\right)\boldsymbol{c}_i^{\perp} = -\left(\boldsymbol{e}_i\times\boldsymbol{e}_{i+1}\right)\boldsymbol{c}_i^{\perp} \\
\nabla\left(\boldsymbol{e}_i\times\boldsymbol{e}_{i+1}\right) &= e_{(i+1)[1]}\nabla e_{i[0]} + e_{i[0]}\nabla e_{(i+1)[1]} - e_{(i+1)[0]}\nabla e_{i[1]} - e_{i[1]}\nabla e_{(i+1)[0]} \\
&= e_{(i+1)[1]}\frac{e_{i[1]}}{r_i}\boldsymbol{e}_i^{\perp} - e_{i[0]}\frac{e_{(i+1)[0]}}{r_{i+1}}\boldsymbol{e}_{i+1}^{\perp} + e_{(i+1)[0]}\frac{e_{i[0]}}{r_i}\boldsymbol{e}_i^{\perp} - e_{i[1]}\frac{e_{(i+1)[1]}}{r_{i+1}}\boldsymbol{e}_{i+1}^{\perp} \\
&= e_{i[0]}e_{(i+1)[0]}\left(\frac{1}{r_i}\boldsymbol{e}_i^{\perp} - \frac{1}{r_{i+1}}\boldsymbol{e}_{i+1}^{\perp}\right) + e_{i[1]}e_{(i+1)[1]}\left(\frac{1}{r_i}\boldsymbol{e}_i^{\perp} - \frac{1}{r_{i+1}}\boldsymbol{e}_{i+1}^{\perp}\right) \\
&= \left(e_{i[0]}e_{(i+1)[0]} + e_{i[1]}e_{(i+1)[1]}\right)\boldsymbol{c}_i^{\perp} = \left(\boldsymbol{e}_i\cdot\boldsymbol{e}_{i+1}\right)\boldsymbol{c}_i^{\perp}
\end{aligned}$$

where

$$\boldsymbol{c}_i = \frac{1}{r_i}\boldsymbol{e}_i - \frac{1}{r_{i+1}}\boldsymbol{e}_{i+1}$$

So for

- Wachspress

$$\begin{aligned}
\boldsymbol{\nabla} w_i^{WP} &= 2\left(\boldsymbol{n}_{i-1}\times\boldsymbol{n}_i\right)\nabla\frac{1}{h_{i-1}h_i} = -2\left(\boldsymbol{n}_{i-1}\times\boldsymbol{n}_i\right)\frac{1}{\left(h_{i-1}h_i\right)^2}\left(-h_i\boldsymbol{n}_{i-1} - h_{i-1}\boldsymbol{n}_i\right) = \\
&= w_i^{WP}\left(\frac{1}{h_{i-1}}\boldsymbol{n}_{i-1} + \frac{1}{h_i}\boldsymbol{n}_i\right) \\
\boldsymbol{R}_i^{WP} &\equiv \frac{\boldsymbol{\nabla} w_i^{WP}}{w_i^{WP}} = \frac{1}{h_{i-1}}\boldsymbol{n}_{i-1} + \frac{1}{h_i}\boldsymbol{n}_i \\
\nabla N_i^{WP} &= \nabla\left(\frac{w_i^{WP}}{\sum_{j=0}^{n-1} w_j^{WP}}\right) = \frac{\boldsymbol{\nabla} w_i^{WP}}{\sum_{j=0}^{n-1} w_j^{WP}} - \frac{w_i^{WP}\boldsymbol{\nabla}\left(\sum_{j=0}^{n-1} w_j^{WP}\right)}{\left(\sum_{k=0}^{n-1} w_k^{WP}\right)^2} = \\
&= \frac{w_i^{WP}}{\sum_{j=0}^{n-1} w_j^{WP}}\frac{\boldsymbol{\partial} w_i^{WP}}{w_i^{WP}} - \frac{w_i^{WP}}{\sum_{k=0}^{n-1} w_k^{WP}}\sum_{j=0}^{n-1}\frac{w_j^{WP}}{\sum_{k=0}^{n-1} w_k^{WP}}\frac{\boldsymbol{\partial} w_j^{WP}}{w_j^{WP}} = \\
&= N_i^{WP}\left(\boldsymbol{R}_i^{WP} - \sum_{j=0}^{n-1} N_j^{WP}\boldsymbol{R}_j^{WP}\right)
\end{aligned}$$

- Mean Value

$$\begin{aligned}
\boldsymbol{\nabla} w_i^{MV} &= \nabla\left(\frac{t_{i-1}+t_i}{r_i}\right) = \frac{\nabla\left(t_{i-1}+t_i\right)}{r_i} - \frac{\left(t_{i-1}+t_i\right)\nabla r_i}{r_i^2} = \\
&= \frac{\nabla\left(t_{i-1}+t_i\right)}{r_i} + \frac{\left(t_{i-1}+t_i\right)}{r_i^2}\boldsymbol{e_i} = \frac{t_{i-1}}{r_i\sin(\alpha_{i-1})}\boldsymbol{c_{i-1}^{\perp}} + \frac{t_i}{r_i\sin(\alpha_i)}\boldsymbol{c_i^{\perp}} + w_i^{MV}\frac{1}{r_i}\boldsymbol{e_i} \\
\boldsymbol{R}_i^{MV} &\equiv \frac{\boldsymbol{\nabla} w_i^{MV}}{w_i^{MV}} = \frac{t_{i-1}}{\left(t_{i-1}+t_i\right)\sin(\alpha_{i-1})}\boldsymbol{c}^{\perp}{}_{i-1} + \frac{t_i}{\left(t_{i-1}+t_i\right)\sin(\alpha_i)}\boldsymbol{c}^{\perp}{}_i + \frac{1}{r_i}\boldsymbol{e}_i \\
\partial N_i^{MV} &= N_i^{MV}\left(\boldsymbol{R}_i^{MV} - \sum_{j=0}^{n-1} N_j^{MV}\boldsymbol{R}_j^{MV}\right)
\end{aligned}$$

3.2 Attributes

3.2.1 Attributes of Wachspress coordinates

Wachspress coordinates used to be the only known type of GBC for a long time. Their strengths are being applicable to all convex polygons (a convex polygon is one with all inner angles lesser or equal to 180°) and being rational functions.

We will firstly prove that the Wachspress coordinates are barycentric. We can see that $\sum_{i=0}^{n-1} N_i(\boldsymbol{v}) = 1$ because $N_i(\boldsymbol{v}) = \frac{w_i}{\sum_{j=0}^{n-1} w_j}$. What is left is to show that

$$\sum_{i=0}^{n-1} N_i(\boldsymbol{v})(\boldsymbol{v}_i - \boldsymbol{v}) = 0 \tag{3.7}$$

Let's choose a fixed $\boldsymbol{v} \in \mathscr{P}$, $\boldsymbol{v}$ can be expressed as a barycentric combination of $\boldsymbol{v}_{i-1}, \boldsymbol{v}_i, \boldsymbol{v}_{i+1}$ as:

$$\begin{aligned}
\boldsymbol{v} &= \frac{A_i}{C_i}\boldsymbol{v}_{i-1} + \frac{C_i - A_{i-1} - A_i}{C_i}\boldsymbol{v}_i + \frac{A_{i-i}}{C_i}\boldsymbol{v}_{i+1} \\
\frac{C_i}{A_{i-1}A_i}\boldsymbol{v} &= \frac{1}{A_{i-1}}\boldsymbol{v}_{i-1} + \frac{C_i}{A_{i-1}A_i}\boldsymbol{v}_i - \frac{1}{A_i}\boldsymbol{v}_i - \frac{1}{A_{i-1}}\boldsymbol{v}_i + \frac{1}{A_i}\boldsymbol{v}_{i+1} \\
\frac{C_i}{A_{i-1}A_i}(\boldsymbol{v}_i - \boldsymbol{v}) &= \frac{1}{A_{i-1}}(\boldsymbol{v}_i - \boldsymbol{v}_{i-1}) - \frac{1}{A_i}(\boldsymbol{v}_{i+1} - \boldsymbol{v}_i)
\end{aligned} \tag{3.8}$$

with A_i and C_i being the same as in Figure 3.2. (3.8) holds no matter if $\boldsymbol{v}$ is inside or outside the triangle with vertices $\boldsymbol{v}_{i-1}, \boldsymbol{v}_i, \boldsymbol{v}_{i+1}$. When we take a sum of both sides from 0 to $n-1$ we get

$$\sum_{i=0}^{n-1}\frac{C_i}{A_{i-1}A_i}(\boldsymbol{v}_i - \boldsymbol{v}) = \frac{1}{A_{n-1}}(\boldsymbol{v}_0 - \boldsymbol{v}_{n-1}) - \frac{1}{A_0}(\boldsymbol{v}_1 - \boldsymbol{v}_0) + \frac{1}{A_0}(\boldsymbol{v}_1 - \boldsymbol{v}_0) - \ldots - \frac{1}{A_{n-1}}(\boldsymbol{v}_0 - \boldsymbol{v}_{n-1}) = 0$$

proving (3.7). We can also see this hold in Figure 3.3 where we plotted $\sum_{i=0}^{n-1} N_i(\boldsymbol{v})$ and $\sum_{i=0}^{n-1} N_i(\boldsymbol{v})\boldsymbol{v}_i$ for a polygon with vertices $(0,0)$, $(4,0)$, $(3,4)$, $(1,4)$.

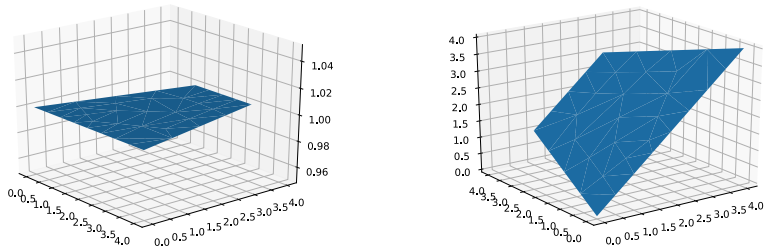

Figure 3.3: Plots of $\sum_{i=0}^{n-1} N_i(\boldsymbol{v})$ and $\sum_{i=0}^{n-1} N_i(\boldsymbol{v})\boldsymbol{v}_i$.

To show that the Wachspress coordinates are indeed rational functions we first express the coordinates in the form

$$N_i(\boldsymbol{v}) = \frac{\hat{w}_i(\boldsymbol{v})}{\sum_{j=0}^{n-1} \hat{w}_j(\boldsymbol{v})}, \quad \hat{w}_i(\boldsymbol{v}) = C_i \prod_{k \neq i-1,i} A_k(\boldsymbol{v})$$

this form can be obtained from (3.3) in the following way

$$(3.3) \implies \frac{w_i^{WP}}{\sum_{j=0}^{n-1} w_j^{WP}} \cdot \frac{\prod_{k=0}^{n-1} A_k}{\prod_{k=0}^{n-1} A_k} = \frac{C_i \prod_{k \neq i-1,i} A_k}{\sum_{j=0}^{n-1} C_j \prod_{k \neq j-1,j} A_k}$$

Now since each area $A_j(\boldsymbol{v})$ is linear for every $\boldsymbol{v}$ it is obvious that $N_i(\boldsymbol{v})$ satisfies the definition of a rational function, that is: every function in the form $\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$ where p, q are polynomials of degrees p_{d_1}, q_{d_2} is a rational function of degree (p_{d_1}, q_{d_2}).

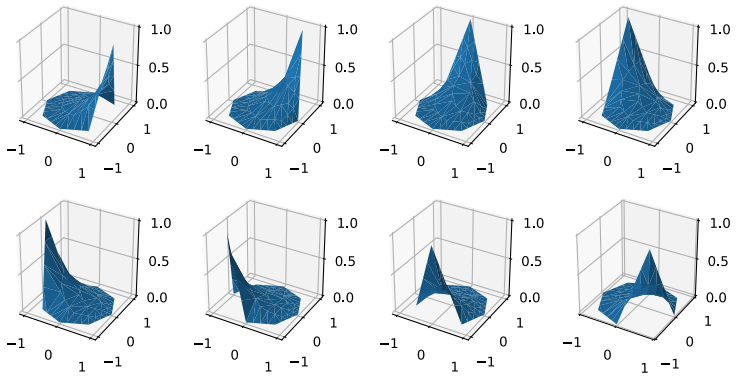

Figure 3.4: Functions N_i for each vertex of an octagon

What the Wachspress coordinates look like In Figure 3.4 we see plots of functions N_i, $i = 0, \ldots, 7$, the polygon over which the functions were computed is a regular octagon with all its

vertices lying on the unit circle.

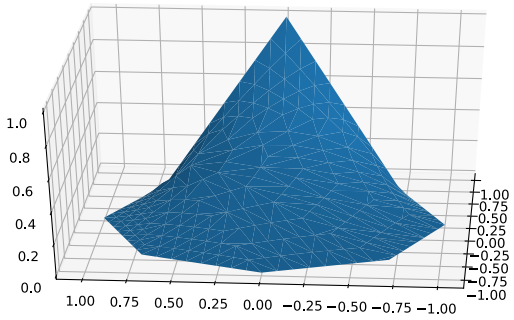

Figure 3.5: Function N_0.

In Figure 3.5 we can see the function N_0 over the same polygon close up, now let's take a look at each of its derivatives.

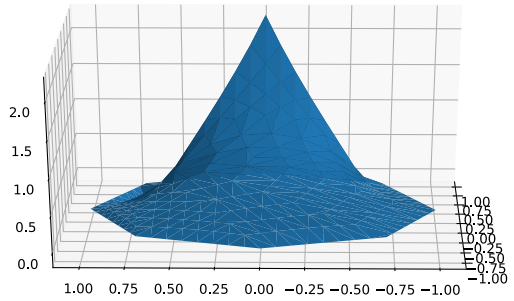

Figure 3.6: Function $\frac{\partial N_0}{\partial x}$.

In Figure 3.6 we can see the derivative of N_0 with respect to x.

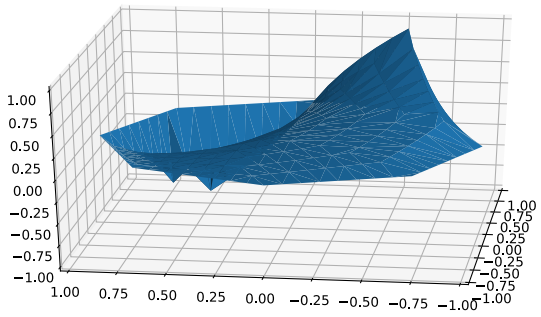

Figure 3.7: Function $\frac{\partial N_0}{\partial y}$.

This last figure shows us the partial derivative of N_0 with respect to y.

3.2.2 Attributes of meanvalue coordinates

While Wachspress coordinates are convenient for their simplicity, they are not well–defined for concave polygons, that is because the denominator in (3.3) can become zero for some points [3]. Meanvalue coordinates on the other hand do have a simple generalization to make them viable for concave polygons, although they generally do not retain positivity in such case. Meanvalue coordinates unlike Wachspress are not rational functions.

The proof that meanvalue coordinates are barycentric can be found in Floater [3].

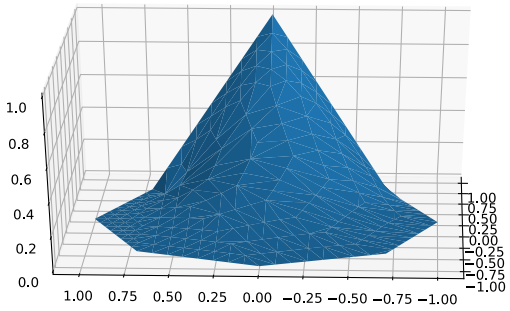

Figure 3.8: Function N_0 with meanvalue coordinates.

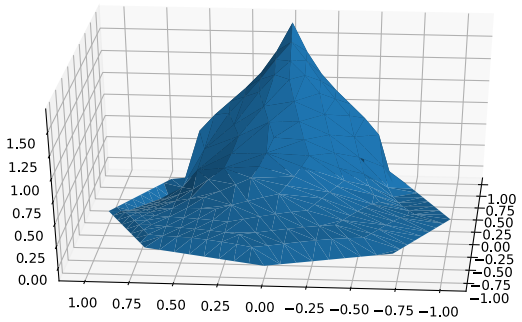

Figure 3.9: Function $\frac{\partial N_0}{\partial x}$ with meanvalue coordinates.

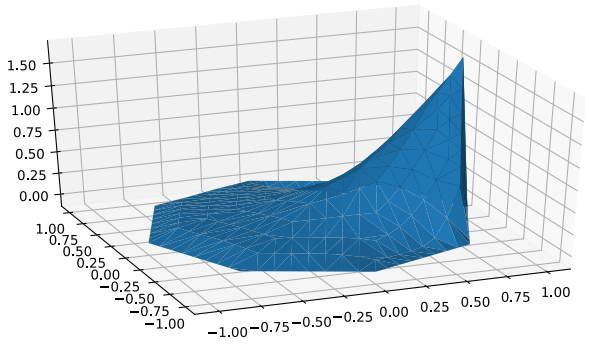

Figure 3.10: Function $\frac{\partial N_0}{\partial x}$ with meanvalue coordinates from the side.

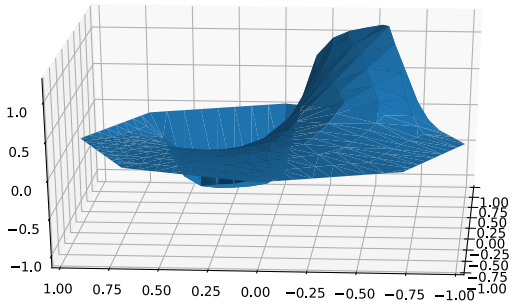

Figure 3.11: Function $\frac{\partial N_0}{\partial y}$ with meanvalue coordinates.

Comparing these figures to their Wachspress counterparts we see that the function N_0 is more curved and seems to start increasing a little sooner. But the difference between their derivatives is more noticeable, the derivative with respect to y is thicker than its counterpart and $\frac{\partial N_0}{\partial x}$ have an upward tendency on the edges, we have not managed to fix this issue. This seems to cause some error when using the meanvalue coordinates, but the error is not big enough to render them unusable.

3.2.3 Attributes of discrete harmonic coordinates

These coordinates are generally not positive, that in turn means not barycentric. The only case when they are positive is when applied to a polygon with all of its vertices lying on a circle. Interestingly enough in this case they are equal to the Wachspress coordinates. For this reason we decided not to use them and not to find their derivatives. For the proof see Floater [4].

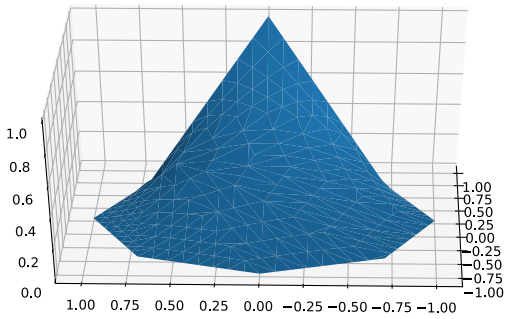

Figure 3.12: Function N_0 with discrete harmonic coordinates.

Definition 17 (Barycentric mapping)**.** Let $\mathscr{P}$ be a polygon with vertices $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n-1}$, $N_i(\boldsymbol{v})$ be the i-th shape function computed over $\mathscr{P}$ and let $\tilde{\mathscr{P}}$ be a polygon with vertices $\tilde{\boldsymbol{v}}_0, \ldots, \tilde{\boldsymbol{v}}_{n-1}$, mapping $\Psi : \mathscr{P} \to \tilde{\mathscr{P}}$ defined as

$$\Psi(\boldsymbol{v}) = \begin{bmatrix} \Psi_1 \\ \Psi_2 \end{bmatrix} (\boldsymbol{v}) = \sum_{i=0}^{n-1} N_i(\boldsymbol{v}) \tilde{\boldsymbol{v}}_i$$

is called a barycentric mapping.

This mapping will be extremely useful for numerical integration because it lets us transform our reference quadrature rules into quadrature rules for integrating over any convex polygon with the same number of vertices. Furthermore, the new quadrature weights can be written as

$\tilde{\boldsymbol{w}} = |J_\Psi|\,\boldsymbol{w}$ where $\boldsymbol{w}$ is a vector of reference quadrature weights and

$$|J_\Psi| = \begin{vmatrix} \partial_{z_1}\Psi_1 & \partial_{z_1}\Psi_2 \\ \partial_{z_2}\Psi_1 & \partial_{z_2}\Psi_2 \end{vmatrix} = \begin{vmatrix} \sum_{i=0}^{n-1} \boldsymbol{v}_{i,1}\partial_{z_1}N_i(\boldsymbol{v}) & \sum_{i=0}^{n-1} \boldsymbol{v}_{i,2}\partial_{z_1}N_i(\boldsymbol{v}) \\ \sum_{i=0}^{n-1} \boldsymbol{v}_{i,1}\partial_{z_2}N_i(\boldsymbol{v}) & \sum_{i=0}^{n-1} \boldsymbol{v}_{i,2}\partial_{z_2}N_i(\boldsymbol{v}) \end{vmatrix} =$$
$$=(\partial_{z_1}\Psi_1)(\partial_{z_2}\Psi_2) - (\partial_{z_1}\Psi_2)(\partial_{z_2}\Psi_1)$$

is the determinant of Jacobian of the barycentric mapping.

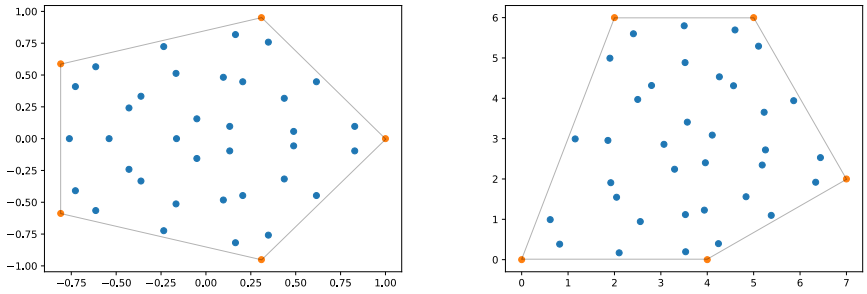

Figure 3.13: Barycentric mapping of a pentagon.

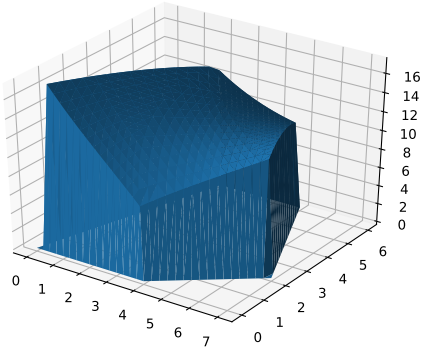

Figure 3.14: $|J_\Psi|$ of the mapping in Figure 3.13 in different $\boldsymbol{v}$.

4 The Finite Element Method

The finite element method (FEM) is a numerical method for solving partial differential equations, it consists of two steps; constructing a discrete approximation and solving it. The discrete approximation will always have the form of a matrix system. For discretizing the PDE (the Poisson equation in our case) we will use the Galerkin's method, but the Rayleigh–Ritz method can also be a viable choice for some PDE's. Galerkin's method assumes, for the purpose of discretization, that all functions of the weak form are from a finite–dimensional vector space spanned by piecewise linear functions that are valued 1 at exactly one node of the mesh and valued 0 at all the other nodes (the term nodes refers to the vertices of elements forming the mesh), this vector space is further a subspace of some Hilbert space (what space depends on the PDE we are solving). Here we mentioned the term mesh which is yet another signature feature of FEM, it is the way we split the whole domain of the PDE, triangular and quadrilateral meshes are most common, but in this thesis we use a polygonal mesh. The PDE we are solving will be valued zero on its boundary, so the discrete system we will construct will be evaluated in all interior nodes of the mesh.

4.1 Deriving FEM

Definition 18 (Poisson equation)**.** The classical form of the Poisson equation modeling heat transfer is

$$-\triangle T = f \qquad \text{in } \Omega, \tag{4.1}$$

$$T = 0 \qquad \text{on } \partial\Omega \tag{4.2}$$

4.1.1 Weak form

Now we will derive the weak form of the Poisson equation, see [5]

$$T, v \in V = H_0^1(\Omega),$$

$$-\int_\Omega (\triangle T) v \,\mathrm{d}V = -\int_\Omega (\nabla^2 T) v \,\mathrm{d}V = \int_\Omega f v \,\mathrm{d}V \tag{4.3}$$

using the identity

$$\boldsymbol{\nabla} \cdot (a\boldsymbol{b}) = \boldsymbol{\nabla} a \cdot \boldsymbol{b} + a(\boldsymbol{\nabla} \cdot \boldsymbol{b})$$

where a is a scalar and $\boldsymbol{b}$ is the term in the integral on the left side of (4.3) becomes

$$-\triangle T v = -v(\boldsymbol{\nabla} \cdot \boldsymbol{\nabla T}) = \boldsymbol{\nabla} v \cdot \boldsymbol{\nabla} T - \boldsymbol{\nabla} \cdot (v \boldsymbol{\nabla} T) \tag{4.4}$$

Applying Gauss divergence theorem

$$\int_\Omega \boldsymbol{\nabla} \cdot \boldsymbol{b} \, \mathrm{d}V = \int_{\partial\Omega} \boldsymbol{n} \cdot \boldsymbol{b} \, \mathrm{d}s \quad \text{(planar Gauss divergence theorem)}$$

where $\int_{\partial\Omega} \boldsymbol{n} \cdot \boldsymbol{b} \, \mathrm{d}s$ is a flux integral of $\boldsymbol{b}$ along $\partial\Omega$, we find that

$$\int_\Omega \boldsymbol{\nabla} \cdot (v \boldsymbol{\nabla} T) \, \mathrm{d}V = \int_{\partial\Omega} \boldsymbol{n} \cdot (v \boldsymbol{\nabla} T) \, \mathrm{d}s = 0, \tag{4.5}$$

this holds because T is defined to be 0 on $\partial\Omega$. We finally get

$$\int_\Omega \left[\boldsymbol{\nabla} v \cdot \boldsymbol{\nabla} T - \boldsymbol{\nabla} \cdot (v \boldsymbol{\nabla} T) \right] \mathrm{d}V = \underbrace{\int_\Omega \boldsymbol{\nabla} v \cdot \boldsymbol{\nabla} T \, \mathrm{d}V}_{a(T,v)} = \underbrace{\int_\Omega f v \, \mathrm{d}V}_{\langle f, v \rangle} \tag{4.6}$$

$$a(T, v) = \langle f, v \rangle$$

4.1.2 Finite elements

Now for the purpose of discretization let's assume that $T, v \in V_h$ with V_h being a vector space such that

$$V_h \subset V, \quad \dim V_h = n \in \mathbb{N}, \quad V_h = \langle \varphi_1, \ldots, \varphi_n \rangle .$$

where φ_i, $i = 1, \ldots, n$ is a piecewise linear function valued 1 in exactly one node of the mesh and valued 0 in all the other nodes.

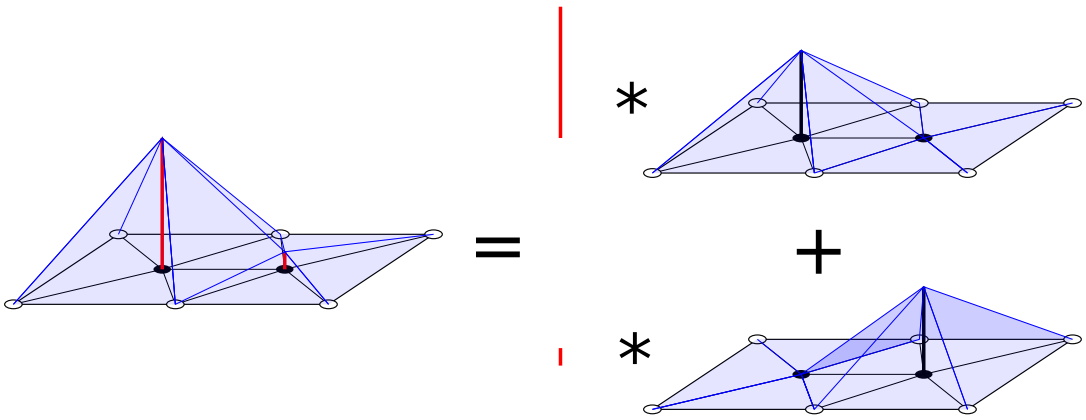

Figure 4.1: Span of V_h.

This means we can rewrite each side of (4.6) as

$$a(T_h, v_h) = \sum_{j=1}^{n}\sum_{i=1}^{n} v_i \underbrace{\left[\int_\Omega (\boldsymbol{\nabla}\varphi_i)\cdot(\boldsymbol{\nabla}\varphi_j)\,\mathrm{d}V\right]}_{a_{ij}} T_j = \boldsymbol{v}^\top \mathsf{A}\boldsymbol{T},$$

$$\langle f, v_h\rangle = \sum_{i=1}^{n} v_i \underbrace{\left[\int_\Omega f\varphi_i\,\mathrm{d}V\right]}_{b_i} = \boldsymbol{v}^\top \boldsymbol{b}$$

where

$$T_h(x,y) = \sum_{j=1}^{n} T_j\varphi_j(x,y),$$
$$v_h(x,y) = \sum_{i=1}^{n} v_i\varphi_i(x,y).$$

It is obvious that we can factor $\boldsymbol{v}^T$ out which leaves us with solving the linear system $\mathsf{A}\boldsymbol{T} = \boldsymbol{b}$. We can assemble A and $\boldsymbol{b}$ using

$$\begin{aligned}
[\mathsf{A}]_{ij} = a_{ij} &= \int_\Omega (\boldsymbol{\nabla}\varphi_i)\cdot(\boldsymbol{\nabla}\varphi_j)\,\mathrm{d}x = \sum_{e=1}^{n_e}\int_{\Omega_e} (\boldsymbol{\nabla}\varphi_i)_{|\Omega_e}\cdot(\boldsymbol{\nabla}\varphi_j)_{|\Omega_e}\,\mathrm{d}x \\
&= \sum_{e=1}^{n_e}\int_{\Omega_e}\left[\boldsymbol{\nabla}(N_I\circ\boldsymbol{\Psi}_e^{-1})\cdot\boldsymbol{\nabla}(N_J\circ\boldsymbol{\Psi}_e^{-1})\right](\boldsymbol{x})\,\mathrm{d}x, \text{ see Figure (4.3)} \\
[\boldsymbol{b}]_i = b_i &= \int_\Omega f(x)\varphi_i\,\mathrm{d}V = \sum_{e=1}^{n_e}\int_{\Omega_e} f\cdot\varphi_i\,\mathrm{d}x \\
&= \sum_{e=1}^{n_e}\int_{\Omega_e}\left[f\cdot(N_I\circ\boldsymbol{\Psi}_e^{-1})\right](\boldsymbol{x})\,\mathrm{d}x,
\end{aligned}$$

where

$$i = \mathrm{loc2glob}_e(I),\ j = \mathrm{loc2glob}_e(J),$$

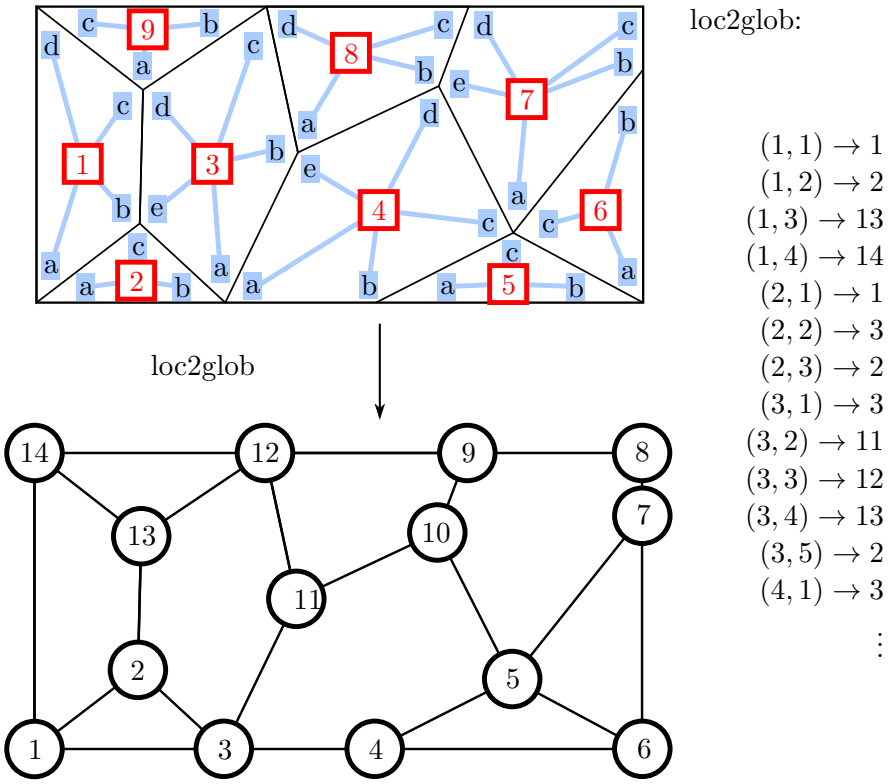

Figure 4.2: The loc2glob function.

We firstly separated Ω into a finite number of elements Ω_e and then used the fact that $\varphi_{i\restriction\Omega_e} = N_I \circ \mathbf{\Psi}_e^{-1}$ illustrated in Figure 4.3.

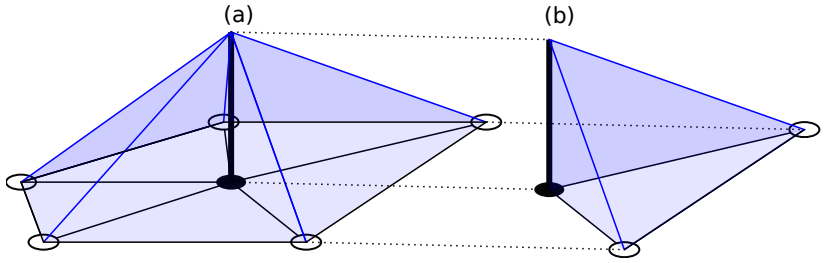

Figure 4.3: φ_i (a) and $N_I \circ \mathbf{\Psi}_e^{-1}$ (b).

Since $\varphi_{i\restriction\Omega_e} = N_I \circ \Psi^{-1}$, N_I itself is the I–th component of the shape function of $\square_e$: $\mathbf{\Psi}_e(\square_e) = \Omega_e$ ($\square_e$ is the reference element with the same number of vertices as Ω_e) , I–th component being the one corresponding to vertex I of element Ω_e. Example of what the graph of function N_I could look like can be found in Figure 3.4.

Now we will use the integration by substitution theorem (4.7) in both the formula for a_{ij}

and b_i

$$\int_{\varphi(U)} f(\boldsymbol{x})\,\mathrm{d}x = \int_U (f \circ \boldsymbol{\varphi})(\boldsymbol{z}) \cdot |\det(D\boldsymbol{\varphi})|\,\mathrm{d}z. \tag{4.7}$$

For the substitution we will use the function $\boldsymbol{\Psi}_e$ such that $\boldsymbol{\Psi}_e : \hexagon_e \to \Omega_e$ is a barycentric mapping from a reference element to an element on the mesh (Definition 17), so

$$\boldsymbol{\Psi}_e(\boldsymbol{z}) = \begin{bmatrix} \Psi_{e,1} \\ \Psi_{e,2} \end{bmatrix} = \sum_{i=1}^{n^e} \boldsymbol{v}_i N_i(\boldsymbol{z})$$

and its Jacobian

$$\mathsf{J}_{\boldsymbol{\Psi}_e}(\boldsymbol{z}) = \begin{bmatrix} \partial_{z_1}\Psi_{e,1} & \partial_{z_1}\Psi_{e,2} \\ \partial_{z_2}\Psi_{e,1} & \partial_{z_2}\Psi_{e,2} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n^e} \boldsymbol{v}_{i,1} (\partial_{z_1} N_i)(\boldsymbol{z}) & \sum_{i=1}^{n^e} \boldsymbol{v}_{i,2} (\partial_{z_1} N_i)(\boldsymbol{z}) \\ \sum_{i=1}^{n^e} \boldsymbol{v}_{i,1} (\partial_{z_2} N_i)(\boldsymbol{z}) & \sum_{i=1}^{n^e} \boldsymbol{v}_{i,2} (\partial_{z_2} N_i)(\boldsymbol{z}) \end{bmatrix}.$$

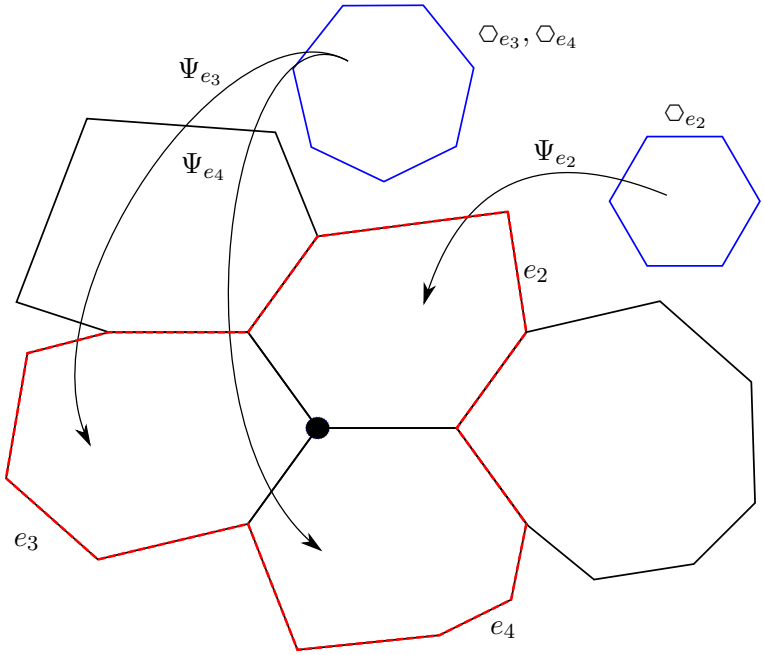

Figure 4.4: Substituting in the mesh.

Applying the substitution we obtain

$$
\begin{aligned}
a_{ij} &= \sum_{e=1}^{n_e} \int_{\square_e} \left(\left[\boldsymbol{\nabla}(N_I \circ \boldsymbol{\Psi}_e^{-1}) \cdot \boldsymbol{\nabla}(N_J \circ \boldsymbol{\Psi}_e^{-1}) \right] \circ \boldsymbol{\Psi}_e \cdot |\det \mathsf{J}_{\boldsymbol{\Psi}_e}| \right) (\boldsymbol{z}) \, \mathrm{d}z \\
&= \sum_{e=1}^{n_e} \int_{\square_e} \left(\left(\begin{bmatrix} \partial_{z_1} N_I \\ \partial_{z_2} N_I \end{bmatrix}^\top \cdot \mathsf{J}_{\boldsymbol{\Psi}_e}^{-\top} \cdot \mathsf{J}_{\boldsymbol{\Psi}_e}^{-1} \cdot \begin{bmatrix} \partial_{z_1} N_J \\ \partial_{z_2} N_J \end{bmatrix} \right) \circ \boldsymbol{\Psi}^{-1} \circ \boldsymbol{\Psi} \cdot |\det \mathsf{J}_{\boldsymbol{\Psi}_e}| \right) (\boldsymbol{z}) \, \mathrm{d}z \\
&= \sum_{e=1}^{n_e} \int_{\square_e} \left(\begin{bmatrix} \partial_{z_1} N_I \\ \partial_{z_2} N_I \end{bmatrix}^\top \cdot \mathsf{J}_{\boldsymbol{\Psi}_e}^{-\top} \cdot \mathsf{J}_{\boldsymbol{\Psi}_e}^{-1} \cdot \begin{bmatrix} \partial_{z_1} N_J \\ \partial_{z_2} N_J \end{bmatrix} \cdot |\det \mathsf{J}_{\boldsymbol{\Psi}_e}| \right) (\boldsymbol{z}) \, \mathrm{d}z \\
&\equiv \sum_{e=1}^{n_e} \sum_{q=1}^{n_{q,e}} \left(\begin{bmatrix} \partial_{z_1} N_I \\ \partial_{z_2} N_I \end{bmatrix}^\top \cdot \mathsf{J}_{\boldsymbol{\Psi}_e}^{-\top} \cdot \mathsf{J}_{\boldsymbol{\Psi}_e}^{-1} \cdot \begin{bmatrix} \partial_{z_1} N_J \\ \partial_{z_2} N_J \end{bmatrix} \cdot |\det \mathsf{J}_{\boldsymbol{\Psi}_e}| \right) (\boldsymbol{z}_q) w_q
\end{aligned}
$$

and

$$
\begin{aligned}
b_i &= \sum_{e=1}^{n_e} \int_{\square_e} \left(\left[f \cdot (N_I \circ \boldsymbol{\Psi}_e^{-1}) \right] \circ \boldsymbol{\Psi}_e \cdot |\det \mathsf{J}_{\boldsymbol{\Psi}_e}| \right) (\boldsymbol{z}) \, \mathrm{d}z \\
&= \sum_{e=1}^{n_e} \int_{\square_e} \left[(f \circ \boldsymbol{\Psi}_e) \cdot N_I \cdot |\det \mathsf{J}_{\boldsymbol{\Psi}_e}| \right] (\boldsymbol{z}) \, \mathrm{d}z \\
&\equiv \sum_{e=1}^{n_e} \sum_{q=1}^{n_{q,e}} \left(f \circ \boldsymbol{\Psi}_e \cdot N_I \cdot |\det \mathsf{J}_{\boldsymbol{\Psi}_e}| \right) (\boldsymbol{z}_q) w_q
\end{aligned}
$$

where we used

$$
\begin{aligned}
\left(N \circ \boldsymbol{\Psi}_e^{-1} \right) (\boldsymbol{x}) &= N \left(\boldsymbol{\Psi}_e^{-1}(\boldsymbol{x}) \right), \quad \boldsymbol{\Psi}_e(\boldsymbol{z}) = \boldsymbol{x} \quad \boldsymbol{z} = \boldsymbol{\Psi}_e^{-1}(\boldsymbol{x}), \\
\boldsymbol{\nabla}_{\boldsymbol{x}} \left(N \circ \boldsymbol{\Psi}_e^{-1} \right) (\boldsymbol{x}) &= \begin{bmatrix} \partial_{x_1} \left[\left(N \circ \boldsymbol{\Psi}_e^{-1} \right) (\boldsymbol{x}) \right] \\ \partial_{x_2} \left[\left(N \circ \boldsymbol{\Psi}_e^{-1} \right) (\boldsymbol{x}) \right] \end{bmatrix} \\
&= \begin{bmatrix} (\partial_{z_1} N) \left(\boldsymbol{\Psi}_e^{-1}(\boldsymbol{x}) \right) \cdot (\partial_{x_1} z_1)(\boldsymbol{x}) + (\partial_{z_2} N) \left(\boldsymbol{\Psi}_e^{-1}(\boldsymbol{x}) \right) \cdot (\partial_{x_1} z_2)(\boldsymbol{x}) \\ (\partial_{z_1} N) \left(\boldsymbol{\Psi}_e^{-1}(\boldsymbol{x}) \right) \cdot (\partial_{x_2} z_1)(\boldsymbol{x}) + (\partial_{z_2} N) \left(\boldsymbol{\Psi}_e^{-1}(\boldsymbol{x}) \right) \cdot (\partial_{x_2} z_2)(\boldsymbol{x}) \end{bmatrix} \\
&= \left(\begin{bmatrix} \partial_{x_1} z_1 & \partial_{x_1} z_2 \\ \partial_{x_2} z_1 & \partial_{x_2} z_2 \end{bmatrix} \cdot \left(\begin{bmatrix} \partial_{z_1} N \\ \partial_{z_2} N \end{bmatrix} \circ \boldsymbol{\Psi}_e^{-1} \right) \right) (\boldsymbol{x}) \\
&= \left(\mathsf{J}_{\boldsymbol{\Psi}_e^{-1}} \cdot \begin{bmatrix} \partial_{z_1} N \\ \partial_{z_2} N \end{bmatrix} \circ \boldsymbol{\Psi}_e^{-1} \right) (\boldsymbol{x}) \\
&= \left(\left(\mathsf{J}_{\boldsymbol{\Psi}_e}^{-1} \cdot \begin{bmatrix} \partial_{z_1} N \\ \partial_{z_2} N \end{bmatrix} \right) \circ \boldsymbol{\Psi}_e^{-1} \right) (\boldsymbol{x}) \\
&= \left(\left(\begin{bmatrix} \partial_{z_1} x_1 & \partial_{z_1} x_2 \\ \partial_{z_2} x_1 & \partial_{z_2} x_2 \end{bmatrix} \cdot \begin{bmatrix} \partial_{z_1} N \\ \partial_{z_2} N \end{bmatrix} \right) \circ \boldsymbol{\Psi}_e^{-1} \right) (\boldsymbol{x})
\end{aligned}
$$

and

$$
\mathsf{J}_{\boldsymbol{\Psi}_e^{-1}}(\boldsymbol{x}) = \left((\mathsf{J}_{\boldsymbol{\Psi}_e})^{-1} \circ \boldsymbol{\Psi}_e^{-1} \right) (\boldsymbol{x})
$$

5 Numerical experiments

In this section we will test the convergence of the method described above as the number of elements comprising the mesh increases, we will perform the test for two functions first of which is $f_1(x,y) = -\triangle \sin(x)\sin(y) = 2\sin(x)\sin(y)$, its analytic solution is clearly $T_1(x,y) = \sin(x)\sin(y)$. This solution satisfies the boundary condition (4.2) on $\Omega_1 = \langle -\pi, \pi\rangle \times \langle -\pi, \pi\rangle$. The second function will be $f_2(x,y) = -2(x^2 + y^2 - 2)$ with analytic solution $T_2(x,y) = (x-1)(x+1)(y-1)(y+1)$, its boundary condition is satisfied on $\Omega_2 = \langle -1, 1\rangle \times \langle -1, 1\rangle$.

For these two experiments we will use a polygonal mesh generated by "clipping" a Voronoi tessellation initialized with pseudo random points. We will not go deeper into the topic of Voronoi tessellations or mesh forming in general as it is not the subject of this thesis. The experiment we will conduct involves solving the Poisson equation using meshes consisting of different ammounts of elements, if every subsequent mesh was completely random a solution using a greater mesh might very well be less exact, so every mesh will be generated with the initialization points of the previous one and the remaining points will be chosen at random (even though it still doesn't comletely guarantee the desired effect). This approach is far from ideal, our sole purpose in choosing it is to demonstrate the ability of our program to deal with "ugly" meshes.

This brings us to the last experiment where we will compare regular (square) meshes with pseudo random meshes.

Experiment one

In Figure 5.1 we can see a plot of $f_1(x,y)$ giving the distribution of heat (generated via Maple)

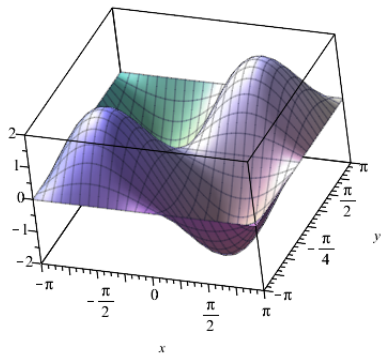

Figure 5.1: $f_1(x,y)$.

Figure **??** depicts the analytic solution for f_1, we can see that its essencially the same function but with half the amplitude.

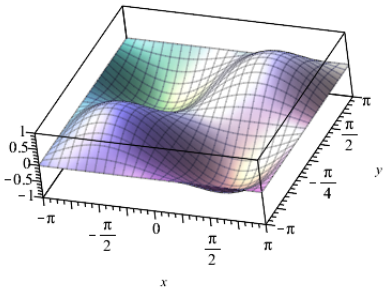

Figure 5.2: Analytic solution for f_1.

Figure 5.3 shows us a numeric solution for f_1 using 3000 elements, here we can see the main drawback of using random meshes, there can appear elements much larger than others ruining the precision.

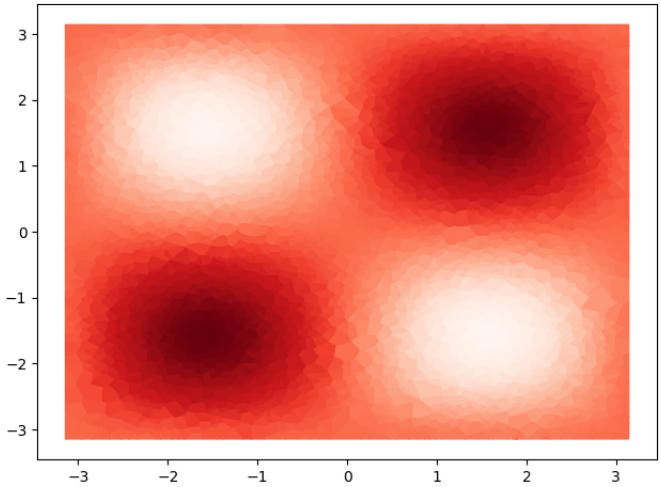

Figure 5.3: Numeric solution for f_1 using a mesh consisting of 3000 elements.

And finally in Figure 5.4 we see the result of this experiment, the decrease of error is clearly logarithmic.

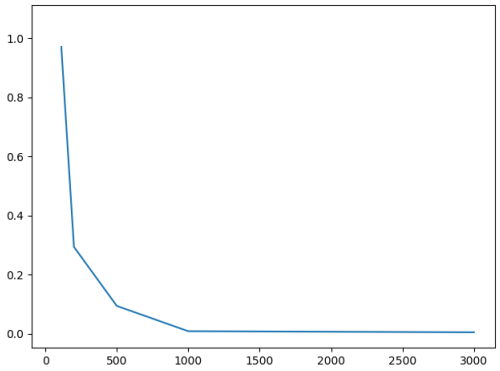

Figure 5.4: Total error as the number of elements comprising the mesh increases.

Experiment two

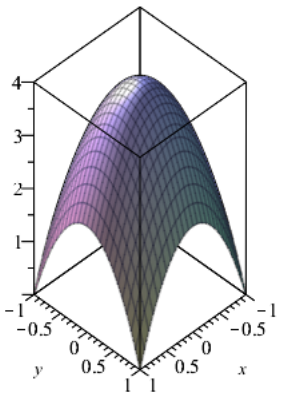

Figure 5.5: $f_2(x, y)$.

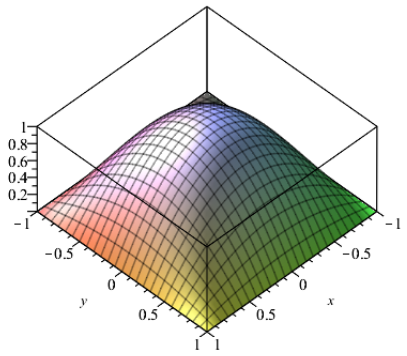

Figure 5.6: Analytic solution for f_1.

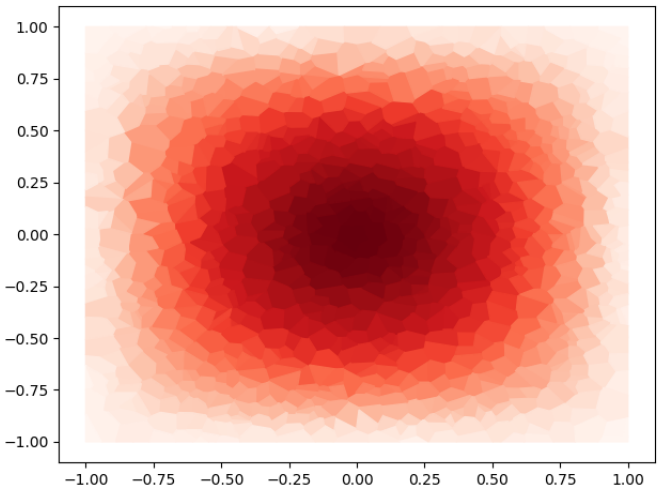

Figure 5.7: Numeric solution for f_2 using a mesh consisting of 1000 elements.

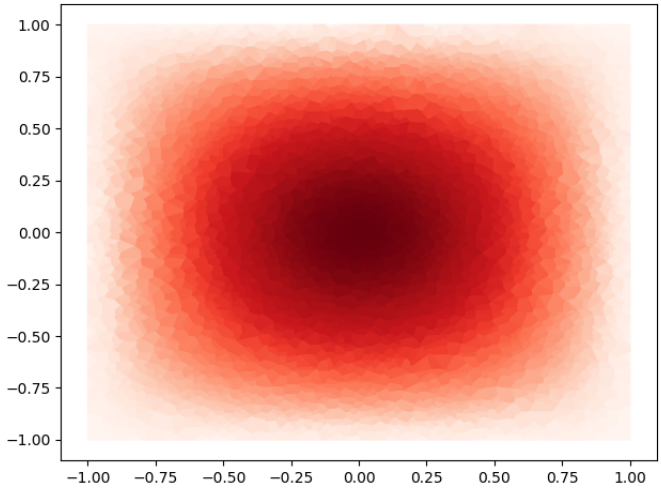

Figure 5.8: Numeric solution for f_2 using a mesh consisting of 3000 elements.

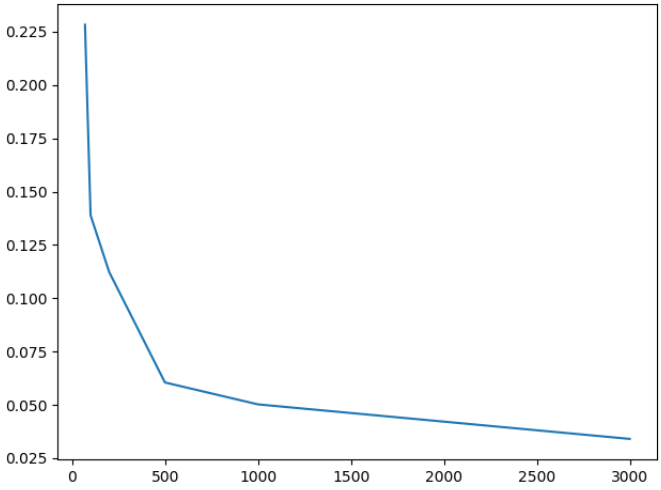

Figure 5.9: Total error as the number of elements comprising the mesh increases.

Comparison of a regular mesh with a pseudorandom mesh

We will compare the convergence of a regular rectagle mesh to that of the one previously used. This test will be done on f_2.

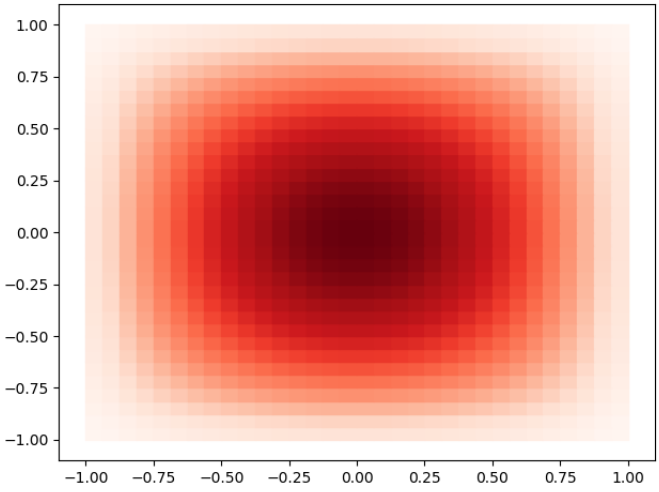

Figure 5.10: Numeric solution for f_2 using a rectangular mesh consisting of 1000 elements.

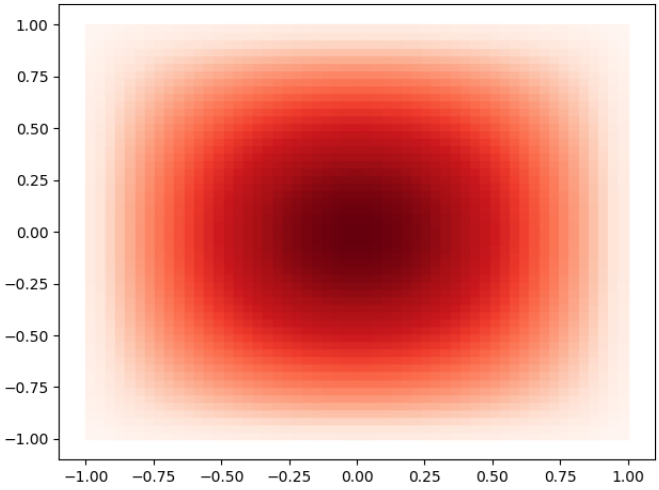

Figure 5.11: Numeric solution for f_2 using a rectangular mesh consisting of 3000 elements.

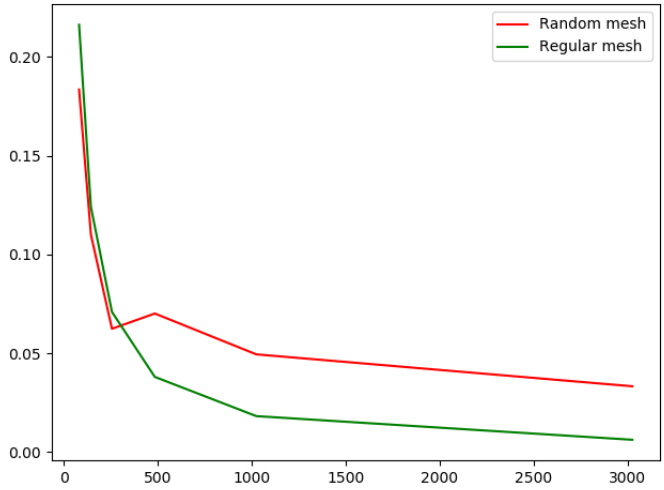

Figure 5.12: Total error as the number of elements comprising the mesh increases.

We can see that when using a regular mesh, the convergence has the same tendency as it did when we used pseudorandom meshes, this justifies our usage of them. However, it is obvious that regular meshes are much better suited for practical uses. There is an upward spike at 500 elements for the random mesh, those can still occur with the way we form the mesh as we have already mentioned, but they will never occur when using the regular rectangular mesh.

6 Conclusion

We have successfully implemented the node elimination algorithm, but as we have already mentioned the algorithm comes with its drawbacks, sometimes the nodes can get moved outside the reference elements, another danger is that the nodes will not be well distributed causing the algorithm to stop early and resulting in a low efficiency quadrature rule. This can be fixed by using different initial quadratures, but usually the algorithm work without any problems.

As for the barycentric coordinates we have decided on using just Wachspress coordinates and Mean Value coordinates because in the case that vertices of reference elements lie on a circle (which is the case for us) the discrete harmonic coordinates are equal to the Wachspress coordinates, of course if the vertices of reference elements did not lie on a circle the coordinates would not be barycentric which is all the more reason to not use them. As can be seen in figures 3.9 and 3.10 the derivative of meanvalue shape functions with respect to x has an upward tendency that should not be there on the edges.

Next we have implemented FEM assemblers of the stiffness matrix and the right side for the Poisson equation and tested their convergence for two different input functions with pseudorandom polygonal meshes and then compared the convergence of a regular and a pseudorandom mesh for one of the previously used functions. The reason why we chose pseudorandom meshes in these two experiments was to show that our implementation can deal with ugly meshes. We have seen that in all cases the total error decreases logarithmically as the number of elements comprising the mesh increases. It is also clear that regular meshes have a better convergence rate that pseudorandom ones.

References

[1] H. Xiao, Z. Gimbutas: A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. Computers and Mathematics with Applications 59(2), 663--676, 2009, doi:10.1016/j.camwa.2009.10.027

[2] M. Floater: Generalized barycentric coordinates and applications. Acta Numerica, 24, 161–214, 2015. doi:10.1017/S0962492914000129

[3] M. Floater: Wachspress and Mean Value Coordinates. Springer Proceedings in Mathematics and Statistics. 83, 81–102, 2014. doi:10.1007/978-3-319-06404-8_6

[4] M.S. Floater, K. Hormann, G. Kós: A general construction of barycentric coordinates over convex polygons. Adv Comput Math 24, 311–331, 2006. doi:10.1007/s10444-004-7611-6

[5] Nonlinear finite elements/Weak form of Poisson equation. (2017, July 26). Wikiversity. Retrieved 00:17, July 26, 2017 [link]